Computers, Materials & Continua

Tech Science Press

Doi:10.32604/cmc.2025.070794

ARTICLE

Lightweight Complex-Valued Neural Network for Indoor Positioning

Le Wang[1], **Bing Xu**[1,*], **Peng Liu**[2] **and En Yuan**[1]

[1]College of Command and Control Engineering, Army Engineering University of PLA, Nanjing, 210007, China
[2]Purple Mountain Laboratories, Nanjing, 211111, China

*Corresponding Author: Bing Xu. Email: bingnj@foxmail.com

Received: 24 July 2025; Accepted: 01 October 2025; Published: 09 December 2025

ABSTRACT: Deep learning has been recognized as an effective method for indoor positioning. However, most existing real-valued neural networks (RVNNs) treat the two constituent components of complex-valued channel state information (CSI) as real-valued inputs, potentially discarding useful information embedded in the original CSI. In addition, existing positioning models generally face the contradiction between computational complexity and positioning accuracy. To address these issues, we combine graph neural network (GNN) with complex-valued neural network (CVNN) to construct a lightweight indoor positioning model named CGNet. CGNet employs complex-valued convolution operation to directly process the original CSI data, fully exploiting the correlation between real and imaginary parts of CSI while extracting local features. Subsequently, the feature values are treated as nodes, and conditional position encoding (CPE) module is applied to add positional information. To reduce the number of connections in the graph structure and lower the model complexity, feature information is mapped to an efficient graph structure through a dynamic axial graph construction (DAGC) method, with global features extracted using maximum relative graph convolution (MRConv). Experimental results show that, on the CTW dataset, CGNet achieves a 10% improvement in positioning accuracy compared to existing methods, while the number of model parameters is only 0.8 M. CGNet achieves excellent positioning accuracy with very few parameters.

KEYWORDS: Indoor positioning; complex-valued neural network; channel state information; lightweight model

1 Introduction

With the development of wireless communications and the widespread adoption of mobile devices, there is an urgent demand for precise location awareness. Location-based services play an important role in many fields and are attracting increasing attention from researchers. Currently, global navigation satellite systems (GNSS), such as GPS, exhibit excellent positioning performance in outdoor environments. However, in indoor environments, GNSS faces the challenge of signal fading, obstruction, and multi-path effects, which severely limit their application in indoor positioning. Traditional indoor positioning methods based on geometric measurements require extremely high clock synchronization and accurate distance and angle measurements [1]. In contrast, CSI-based fingerprinting methods can directly use existing Wi-Fi or 5G network devices without requiring additional positioning infrastructure, making them easier to deploy and use.

CSI reflects the status information of signal transmission between the transmitter and receiver, including channel gain, phase offset, and delay. This information is crucial for achieving high-precision positioning. CSI is in complex-valued form, and existing real-valued neural network (RVNN) based methods are unable

to process complex-valued data. Complex-valued CSI must be converted o real-valued form through: 1) separating the real and imaginary parts [2], 2) using the power of the real and imaginary parts [3], 3) converting to polar domain values [4]. These conversion methods may fail to fully exploit the correlation between the real and imaginary parts of CSI, thereby affecting the accuracy of the positioning algorithm.

CVNN model can utilize both real and imaginary components of complex-valued data, which has been successfully applied in some fields, such as speech processing, automatic modulation classification, images processing. Trabelsi et al. [5] systematically constructed a framework for the complex-valued neural network operation, and verified that CVNN model outperformed RVNN model in music transcription as well as in signal processing. Yu et al. [6] applied CVNN to indoor positioning and achieved good results. However, convolution operations mostly extract shallow local features, so we need to accumulate multiple layers of convolution operations to extract global features, resulting in a high model complexity.

Existing indoor positioning models generally face the conflict between computational complexity and positioning accuracy. Lightweight structure design is an important foundation for achieving model lightweight. We designed a lightweight indoor positioning model that employs complex-valued convolutions for local feature extraction and employs conditional position encoding module to add positional information to each feature node. Subsequently, efficient graph convolutions are employed for global feature extraction. This synergy significantly enhances the model's feature extraction capability while maintaining its lightweight characteristics. Finally, complex-valued fully connected layers process the feature information to perform position prediction. Experimental results demonstrate that CGNet achieves over a 10% improvement in positioning accuracy on the CTW dataset compared to existing methods, with a model parameter count of only 0.8 M.

2 Related Work

2.1 Indoor Positioning Model

Existing research has introduced convolutional neural networks into the field of CSI-based indoor positioning. Cerar et al. [7] proposed the CNN4 series models for large-scale MIMO scenarios, extracting CSI features through stacked real-valued convolutional layers. Zhang et al. [8] proposed AAResCNN, introducing attention-enhanced residual modules to capture fine-grained CSI features. Wan et al. [2] proposed an attention-based CSI positioning method that dynamically adjusts the importance of channel features by analyzing the quality of the channel response, thereby enhancing positioning accuracy. However, converting complex CSI to real CSI may fail to fully leverage the correlation between the real and imaginary parts of CSI, thereby affecting the accuracy of positioning algorithms.

To address the limitation of RVNNs in fully leveraging complex-valued correlations, complex-valued neural networks (CVNNs) are introduced into CSI positioning to enable direct processing of complex-valued CSI, thereby preserving complete channel information. Yu et al. [6] applied CVNN to indoor positioning. By learning complex features of CSI through convolutional and fully connected layers, they achieved a 5%–8% accuracy improvement over RVNN. Xiao et al. [9] proposed CDSCNN, which simplifies the channel-space correlation mapping by decomposing complex-valued convolution into spatial-dimensional depth convolution and channel-dimensional point convolution. However, this approach relies on complex-valued convolution for local feature extraction, resulting in insufficient global feature capture.

GNN improves performance by modeling the relationships between nodes and edges of the data topology, where signal features are modeled as nodes and their interconnections represent the interactions between these features. The core of GNN is to update its feature representation by recursively aggregating the features of neighboring nodes, thereby capturing complex topological information and relationships in the

graph structure. A key aspect of constructing GNN-based models lies in mapping signals to graph structures. Common graph construction algorithms include KNN [10], SVGA [11], MGC [12], etc. KNN captures local feature similarity by selecting the nearest-neighbor nodes, but requires computing and sorting distances to all nodes, resulting in high complexity. SVGA constructs static graphs using fixed axial connections, which is computationally efficient but lacks dynamic adaptation. MGC introduces conditional position coding to enhance the spatial awareness of sparse graphs, but the fixed number of connections limits the flexibility of feature interaction. Cai et al. [13] combined efficient partial convolution and graph sparse attention techniques to convert convolution-extracted features into graph structures, enabling the extraction of more potential features from a graph perspective.

2.2 Lightweight Approach

Recent studies have proposed various lightweight indoor positioning methods, including lightweight structural design, model pruning, knowledge distillation and so on. Wimmer et al. [14] applied weight pruning to eliminate redundant network connections. Lu et al. [15] adopted a two-phase simulation-exploration training strategy to facilitate knowledge transfer between teacher and student models. However, these compression methods often lead to substantial performance degradation. Sun et al. [16] trained a single lightweight network to process the real and imaginary parts of the CSI separately, thereby reducing model complexity but compromising feature correlation between the two components.

3 Proposed Method

3.1 Problem Statement

In a massive MIMO positioning system, CSI indicates the change that the signal experiences while traversing the channel. This change is dependent on the position from which the signal is transmitted. CSI is related to the transmitted signal $T_{i,j}$ and the received signal $R_{i,j}$ at antenna i on subcarrier j, as shown in Eq. (1). N denotes Gaussian white noise. CSI is a complex number, it can be expressed in Cartesian form using Eq. (2), which consists of a real part and an imaginary part.

$$R_{i,j} = T_{i,j} \cdot CSI_{i,j} + N \tag{1}$$

$$CSI_{i,j} = Re + iIm \tag{2}$$

The positioning dataset is constructed by complex-valued CSI and corresponding true spatial coordinates $P(x, y, z)$. We attempt to design an indoor positioning model to learn location-related features in CSI. For a new unknown location with measured CSI, the model can predict the user's estimated spatial coordinates $\hat{P}(\hat{x}, \hat{y}, \hat{z})$. The positioning problem can be formulated as an optimization problem, where the objective is to minimize the discrepancy between the estimated and actual positions of the user, given by

$$\min \frac{1}{K} \sum_{k=1}^{K} \left[\| P_k - \hat{P}_k \|_2^2 \right] \tag{3}$$

where K represents the number of user samples, P_k and $\hat{P}_k$ denote the true and estimated locations of the k-th user, respectively.

3.2 System Model

Currently, CSI-based positioning methods mainly use RVNNs, which require converting complex-valued CSI into real-valued representations to adapt to real-valued machine learning models. However, these

conversion methods may cause loss of information from the original complex features and disregard the correlation between the real and imaginary parts, thus reducing the accuracy of the positioning algorithm. For this reason, we propose a lightweight complex-valued neural network for indoor positioning, which can directly process the original complex-valued CSI data and effectively retain the channel feature information.

The specific architecture of the CGNet model is shown in Fig. 1. The CSI processing workflow in CGNet comprises three stages, collaboratively achieving precise mapping from raw CSI to spatial coordinates. The first stage is the local feature extraction phase, consisting of three complex-valued convolution-based feature extraction(CFE) modules. Each module extracts local features from the raw complex CSI through complex-valued convolution and adds positional information to each feature node through CPE module. Features are then converted into graph-structured data using the DAGC [17] method. Finally, global feature extraction is performed using MRConv [18] from a graph-structured perspective, and the extracted feature information is fed into a complex fully-connected layer for position prediction.

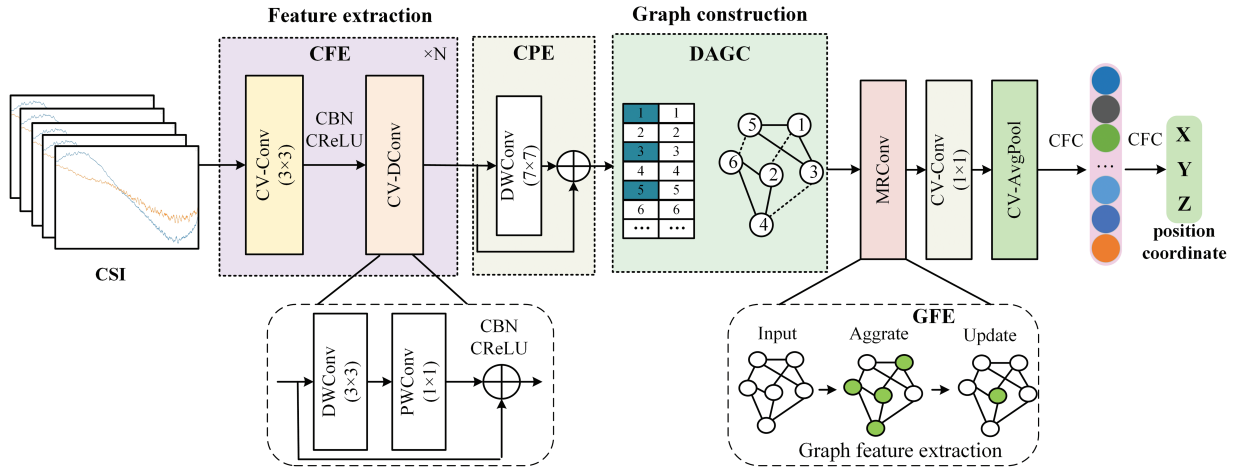


Figure 1: Overall architecture of CGNet

3.3 Complex-Valued Neural Networks

In this section, we present the implementation of the complex-valued building blocks in complex-valued convolutional neural networks, including complex-valued convolution, complex-valued fully connected layers and so on.

1) Complex-Valued Convolution: We define a complex-valued convolution kernel $K = K_r + jK_i$ and a complex-valued input $X = X_r + jX_i$ to represent the CSI, where $j = \sqrt{-1}$. The complex-valued convolution operation can be expressed as:

$$K * X = (K_r * X_r - K_i * X_i) + j(K_i * X_r + K_r * X_i) \tag{4}$$

where $*$ represents the convolution operator. The diagram of the complex-valued convolution operator is shown in Fig. 2.

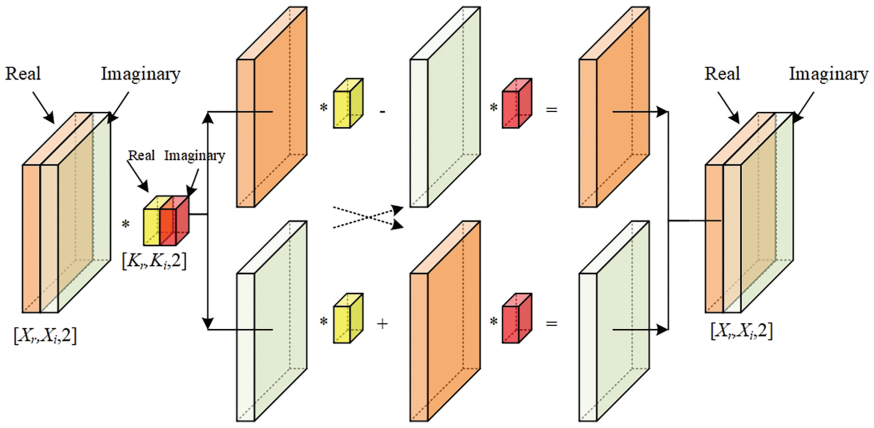


Figure 2: Complex-valued convolution operation

2) Complex-Valued Fully Connected Layer: The computation of the complex-valued fully connected layer (CFC) also conforms to the rules of complex-valued multiplication. The formula is similar to that of complex-valued convolution:

$$CFC = (W_r \cdot X_r - W_i \cdot X_i) + j(W_i \cdot X_r + W_r \cdot X_i) \tag{5}$$

where W_r and W_r denote the weights corresponding to the real and imaginary components of the complex-valued input $X = X_r + jX_i$, respectively. The operator $\cdot$ denotes matrix multiplication.

3) Complex-Valued Activation Function and Pooling Layer: The CReLU function applies the ReLU function to the real and imaginary parts, respectively. The CReLU is defined as:

$$CReLU(X) = ReLU(X_r) + jReLU(X_i). \tag{6}$$

Complex-valued average pooling averages the real and imaginary parts separately. The Complex-valued average pooling is defined as:

$$CAvg(X) = AvgPooling(X_r) + jAvgPooling(X_i). \tag{7}$$

4) Complex-Valued Normalization Layer: For the complex-valued vector X, the batch normalization process is expressed as follows:

$$\tilde{X} = (V)^{-\frac{1}{2}}(X - E[X]) \tag{8}$$

$$CBN(X) = \gamma\tilde{X} + \beta \tag{9}$$

where γ and β are scaling and shifting parameters, respectively. V and $E[X]$ are the covariance matrix and expectation of X, respectively, the covariance matrix V can be expressed as:

$$V = \begin{pmatrix} V_{rr} & V_{ri} \\ V_{ir} & V_{ii} \end{pmatrix} = \begin{pmatrix} \mathrm{Cov}(\mathrm{Re}(X), \mathrm{Re}(X)) & \mathrm{Cov}(\mathrm{Re}(X), \mathrm{Im}(X)) \\ \mathrm{Cov}(\mathrm{Im}(X), \mathrm{Re}(X)) & \mathrm{Cov}(\mathrm{Im}(X), \mathrm{Im}(X)) \end{pmatrix} \tag{10}$$

where $Re(X)$ and $Im(X)$ are the real and imaginary parts of the input data X, respectively.

3.4 Complex-Valued Convolution-Based Feature Extraction

The CFE module consists of complex-valued convolution operations, including complex-valued convolution (CV-Conv) and complex-valued depthwise separable convolution (CV-DConv). The main function is to extract the local features from CSI data. CV-DConv employs a complex-valued convolution kernel with size 3×3 to directly process the complex-valued data, preserving both real and imaginary components while adjusting the feature dimensions. Composed of DWConv and PWConv, CV-DConv fuses information between the real and imaginary parts and further extracts local features. Since DWConv operates only on a subset of channels, PWConv is subsequently used to integrate information across all channels, thereby enhancing information interaction between the real and imaginary parts.

Assume the input complex-valued feature map has $2N$ channels, with N real and N imaginary channels. The convolution kernel size is $K \times K$, and the output complex-valued map has a shape of $2M \times H \times W$, where $H \times W$ denotes its size, $2M$ corresponds to M real channels and M imaginary channels. For floating-point computation volume, each multiplication or addition is counted as a single floating-point operation. A single complex-valued convolution operation performs four convolution multiplications and two element-wise additions, with the number of parameters and floating-point computation, respectively:

$$N_{params}^{CV-Conv} = 2 \times K^2 \times N \times M \tag{11}$$

$$\begin{aligned} N_{FLOPs}^{CV-Conv} &= \left[\left(N \times K^2\right) + \left(N \times K^2 - 1\right)\right] \times M \times H \times W \times 4 + M \times H \times W \times 2 \\ &\approx 8 \times N \times K^2 \times M \times H \times W + 2 \times M \times H \times W \end{aligned} \tag{12}$$

In contrast, CV-DConv simplifies the mapping of channel and spatial correlations, reducing redundant features in convolution operations to improve performance. It first performs channel convolution in the spatial dimension, where each input channel is processed independently using real-valued convolution kernels. Subsequently, complex-valued pointwise convolution is applied to the feature maps output by DWConv. The total number of parameters and floating-point computations are as follows:

$$N_{params}^{CV-DConv} = N_{params}^{DWConv} + N_{params}^{PWConv} = 2 \times K^2 \times N + 2 \times M \times N \tag{13}$$

$$\begin{aligned} N_{FLOPs}^{CV-DConv} &= N_{params}^{DWConv} + N_{params}^{PWConv} = \left[K^2 + \left(K^2 - 1\right)\right] \times 2N \times H \times W + \left[N + \left(N - 1\right)\right] \\ &\quad \times M \times H \times W \times 4 + M \times H \times W \times 2 \\ &\approx 4 \times K^2 \times N \times H \times W + 8 \times N \times M \times H \times W + 2 \times M \times H \times W \end{aligned} \tag{14}$$

Therefore, the total number of parameters and floating-point computation of the CFE module can be expressed as:

$$N_{params}^{CFE} = N_{params}^{CV-Conv} + N_{params}^{CV-DConv} \tag{15}$$

$$N_{FLOPs}^{CFE} = N_{params}^{CV-Conv} + N_{params}^{CV-DConv} \tag{16}$$

3.5 Graph Convolution-Based Feature Extraction

By modeling signal features as nodes and their interconnections as interactions between these features. Graph neural network can provide an in-depth analysis of the complex relationships between signal features. DAGC builds upon SVGA by dynamically generating masks based on Manhattan distance and adaptively filtering connections between similar nodes, achieving both dynamic adaptability and efficiency. Before

graph construction, spatial location-aware encoding is generated by CPE module. This module adopts a grouped convolutional structure to maintain channel independence. Positional features are generated via convolution operations and then element-wise summed with the original input features, providing differentiated spatial feature representations for regional nodes.

MobileViG [11] proves that not all patch needs to be considered, so connections across rows and columns are selected to reduce computational complexity. To reduce the number of comparisons, DAGC leverages the axial construction to improve the construction efficiency by dividing the feature map into four quadrants and calculating the Manhattan distances between the original map and the diagonally flipped quadrants, indirectly estimating the global statistics μ and σ. The reason for using μ and σ is that node pairs within one σ of μ are close to each other and should share information. If the distance between two nodes is less than the estimated difference between μ and σ then connect both nodes.

After obtaining μ and σ, an axial scrolling operation is performed along the height and width dimensions of the feature map. Valid connections are filtered by computing the Manhattan distance of neighboring window features: if the node distance is less than the dynamic threshold $\mu - \sigma$, the connections are retained, and neighborhood difference features are aggregated, maximizing the accumulation of such features. Finally, the original features and dynamically aggregated neighborhood features are spliced along the channel dimension and output after dimensionality reduction by 1×1 convolutional fusion. During graph construction, feature values remain unchanged, and only a fixed graph structure is used to connect them, which ensures information integrity. Therefore, the local information extracted by CFE is completely retained in the GFE module.

During the process of graph construction, MRConv is used to enhance features by maximizing the feature differences between nodes and their neighbors. This module decomposes traditional graph convolution into two phases: feature aggregation and feature update:

$$\mathcal{G}' = \mathcal{F}(\mathcal{G}, \mathcal{W}) = U\left(f(\mathcal{G}, \mathcal{W}_{\text{agg}}), \mathcal{W}_{\text{update}}\right) \tag{17}$$

where $\mathcal{W}_{\text{agg}}$ and $\mathcal{W}_{\text{update}}$ are the learnable weights for aggregation and update operations, respectively. The update of a node is computed by aggregating the features of all connected nodes v_{ij} in v. The aggregation operation and update operation can be represented as:

$$f(\cdot) = [v_{ij}, \max\{v_{i+nk,j+nk} - v_{ij} | i + nk, j + nk \in \mathcal{N}(v_{ij})] \tag{18}$$

$$U(\cdot) = v_{ij}\mathcal{W}_{\text{update}} \tag{19}$$

where n is a parameter controlling the distance between each node, v_{ij} denotes a single node, and $\mathcal{N}(v_{ij})$ is the neighboring node set of v_{ij}. To enhance the feature expressiveness and better capture complex relationships between the nodes, node features are updated using learnable parameters after aggregation. The update operation $h(\cdot)$ employs a 1×1 convolution fully connected layer to update the parameters of aggregated features. Specific implementation details are provided in Algorithm 1.

Algorithm 1: DAGC with MRConv

Given: K_h, K_w: the distance between connections; H, W: feature size; X: input feature map; X_{flipped}: feature map after flip; m: the distance of each roll; u: mean; σ: standard deviation

1: $m \leftarrow 0, u \leftarrow \text{mean}(X, X_{\text{flipped}}), \sigma \leftarrow \text{std}(X, X_{\text{flipped}})$ ▷ Initialize parameters
2: **while** $mK_h < H$ **do** ▷ Loop scrolling
3: $X_{\text{rolled}} \leftarrow \text{roll}_{\text{down}}(X, mK)$

(Continued)

Algorithm 1 (continued)

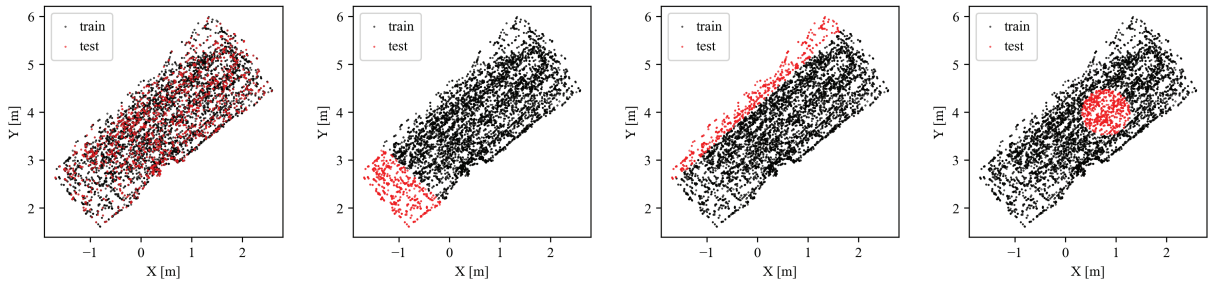
if $\text{dist}(X, X_{\text{rolled}}) < u - \sigma$ **then** ▷ Generate mask
  mask $\leftarrow 1$
else
  mask $\leftarrow 0$
end if
$X_{\text{down}} \leftarrow \text{mask} * (X_{\text{rolled}} - X)$ ▷ Get features
$X_j \leftarrow \max(X_{\text{down}}, X_j)$ ▷ Keep the maximum
$m \leftarrow m + 1$
end while
$m \leftarrow 0$
while $mK_w < W$ **do** ▷ Loop scrolling
$X_{\text{rolled}} \leftarrow \text{roll}_{\text{right}}(X, mK)$
if $\text{dist}(X, X_{\text{rolled}}) < u - \sigma$ **then** ▷ Generate mask
  mask $\leftarrow 1$
else
  mask $\leftarrow 0$
end if
$X_{\text{right}} \leftarrow \text{mask} * (X_{\text{rolled}} - X)$ ▷ Get features
$X_j \leftarrow \max(X_{\text{right}}, X_j)$ ▷ Keep the maximum
$m \leftarrow m + 1$
end while
return $\text{Conv2d}(\text{Concat}(X, X_j))$

4 Results and Discussion

4.1 Dataset and Experimental Environment Configuration

We use the publicly available indoor positioning dataset CTW [19] and KU Leuven dataset [20], provided by the IEEE Communications Theory Workshop, for experimental validation. The CTW dataset is collected using a massive MIMO channel sounder, which measures CSI data of the channel frequency response between a moving transmitter and a fixed receiver with an 8×2 antenna array in a 4×2 meter indoor environment. The transmission frequency is 1.25 GHz with a 20 MHz bandwidth, using 1024 subcarriers. 100 subcarriers are used as guard bands. The total number of samples is 17,486, where each sample consists of CSI and corresponding position coordinates.

To ensure the unbiasedness and generalization ability in model evaluation and avoid the impact of training data selection on results, the data partitioning method from [21] is used to divide the CTW dataset into four different training and testing sets: (1) random training and testing region partitioning; (2) long-edge split with a narrow evaluation area; (3) short- edge split with a wide evaluation area; and (4) cut-out split with testing area within the training area. A two-dimensional visualization of the four methods is shown in Fig. 3. The training set is labeled in black and the testing set in red. All methods split the dataset into training and testing subsets in a ratio of 9:1 and are trained on NVIDIA GeForce RTX 4090 GPU with 500 epochs and a batch size of 32.



(a) Random **(b)** Narrow **(c)** Wide **(d)** Within

Figure 3: Four types of training and testing subset division

The KU Leuven dataset contains CSI data recorded in 9 m^2 indoor environments using the KU Leuven Massive MIMO test platform. Based on different antenna configurations at the base station, the data is categorized into: Uniform Rectangular Array (URA) with 8×8 antennas and Distributed Uniform Linear Array (DIS) with 1×8 antennas and Uniform Linear Array (ULA) with 1×64 antennas. The total number of samples is 252,004. In the KU Leuven dataset, 90% of the samples are randomly allocated to the training set, while the remaining 10% are assigned to the test set. Other training parameters are configured identically to the CTW dataset.

Performance evaluation is conducted using mean distance error (MDE) and root mean squared error (RMSE), which are calculated by (20) and (21). MDE represents the average distance error between the true and estimated positions. RMSE denotes the square root of the average squared difference between the true and the estimated positions. Both metrics are expressed in meters.

$$MDE = E\left[\| p - \hat{p} \|_2\right] \tag{20}$$

$$RMSE = \sqrt{E\left[\| p - \hat{p} \|_2^2\right]} \tag{21}$$

where p and $\hat{p}$ denote the true position and the estimated position.

4.2 Comparison Experiments

Table 1 presents a comparison between CGNet and other related models. Existing positioning models typically process CSI by converting its real and imaginary parts to the real-valued domain for prediction. By contrast, CGNet, which employs a complex-valued convolutional network, exhibits superior performance in both RMSE and MDE metrics. Under random distribution, compared to ACPNet, CGNet further improves RMSE and MDE by 14.1% and 6.0%. For nonuniform distributions, the significant distribution discrepancy between training and testing datasets results in relatively low localization accuracy. The narrow distribution has the worst performance, with the best positioning error MDE of only 1.220 m. In scenarios with slightly balanced training distributions and wide evaluation areas, our proposed model achieves better results compared to other real-valued models.

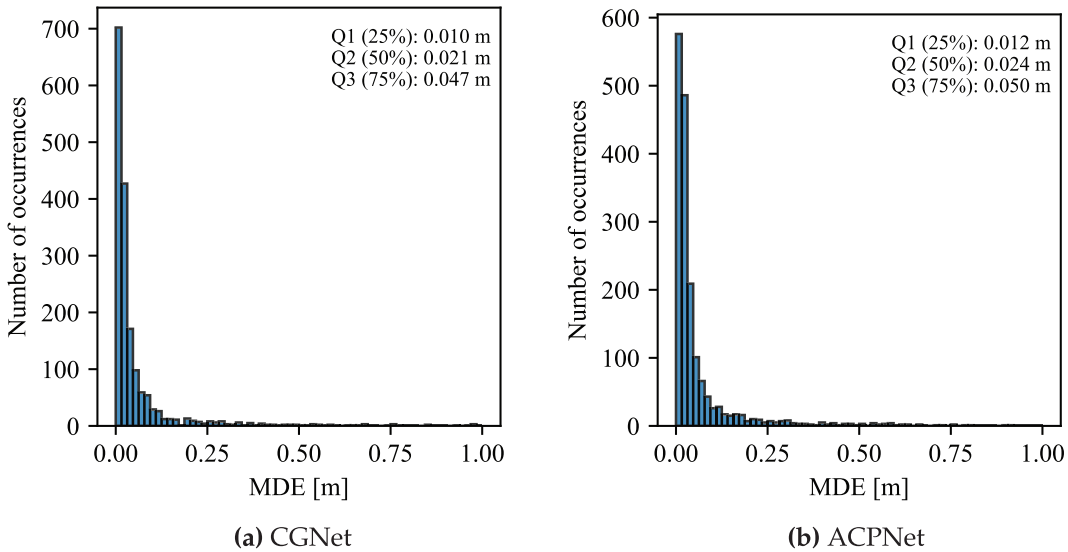
In terms of system resource overhead, parameter, floating-point operations (FLOPs) and inference time are commonly used to evaluate the memory and computational requirements of models. Specific comparisons are provided in Table 1, which show that CGNet reduces parameters and FLOPs by 75.4% and 84.1%, respectively, compared to ACPNet, which has the highest positioning accuracy. By converting features into graph structures, CGNet extracts more latent features from a graph-theoretic perspective. Additionally, the adoption of dynamic graph construction minimizes the computational cost of mapping features to

graph structures, achieving a favorable balance between performance and complexity. The inference time is calculated by feeding test samples sequentially to the network, and finally averaging the inference times of the test samples. CGNet achieves a single-sample inference time of just 4.3 ms, making it the fastest among all compared models. In summary, CGNet achieves optimal positioning accuracy while effectively reducing resource overhead.

Table 1: Performance comparison of different models on CTW dataset

Approach	Random		Within		Narrow		Wide		Complexity		
	RMSE (m)	MDE (m)	RMSE (m)	MDE (m)	RMSE (m)	MDE (m)	RMSE (m)	MDE (m)	Params (M)	FLOPs (M)	Inference Time (ms/sample)
Chin [22]	0.100	0.093	0.381	0.620	0.854	1.326	0.530	0.808	13.7	328.09	/
CNN4 [7]	0.122	0.149	0.365	0.552	0.819	1.286	0.514	0.787	5.3	/	/
CNN4R [7]	0.113	0.127	0.351	0.521	0.776	1.227	0.539	0.835	10.8	/	/
CNN4S [7]	0.108	0.120	0.351	0.524	0.821	1.285	0.528	0.804	16.3	/	/
PirnatEco [21]	0.109	0.112	0.398	0.596	0.801	1.260	0.523	0.793	3.06	174.64	6.0
AAResCNN [8]	0.091	0.088	0.399	0.605	0.854	1.363	0.527	0.795	3.12	174.06	7.9
PKINet [23]	0.087	0.071	0.407	0.608	0.780	1.234	0.531	0.803	2.71	160.44	12.5
MogaNet [24]	0.087	0.070	0.382	0.577	0.848	1.346	0.527	0.798	2.84	154.59	9.8
Transformer [25]	0.088	0.072	0.402	0.581	0.841	1.326	0.526	0.801	2.95	298.00	18.2
CVNN [5]	0.080	0.062	0.388	0.569	0.840	1.339	0.532	0.801	1.66	174.63	14.0
ACPNet [2]	0.085	0.066	0.389	0.596	0.840	1.335	0.532	0.806	3.26	194.86	13.5
CGNet	0.073	0.062	0.349	0.516	0.770	1.220	0.502	0.769	0.8	30.98	4.3

To further highlight the performance advantages of CGNet, Fig. 4 shows the histograms of error distribution for the random scenario. Compared with the real-valued model ACPNet, the MDE distribution obtained by the complex-valued model proposed in this paper is narrower around very small values. Half of the data points exhibit an MDE within 0.021 m, significantly lower than ACPNet. This demonstrates the superiority of our proposed model in dealing with CSI.



(a) CGNet **(b)** ACPNet

Figure 4: Histograms of the error distribution (**a**) The result of the CGNet (**b**) The result of the ACPNet

To validate the generalization capability of the CCPNet model, the Ku Leuven University dataset was introduced as a universal experimental dataset. Specific experimental results are shown in Table 2. The results demonstrate that CGNet exhibits outstanding performance in the URA, ULA, and DIS scenarios,

achieving MDE of 0.0061, 0.0049, and 0.0039 m, respectively. CGNet achieves high-precision positioning through efficient feature modeling using complex architecture and graph convolutions. This approach fully preserves the correlation between the real and imaginary parts of CSI. These results demonstrate that CGNet can more accurately capture positional features across diverse antenna deployment scenarios, exhibiting strong adaptability.

Table 2: Performance comparison of different models on KU Leuven dataset

Approach	URA	ULA	DIS
	MDE (m)	MDE (m)	MDE (m)
MPCs-based [20]	0.0916	0.0578	0.0163
AAResCNN [8]	0.0261	0.0162	0.0141
PKINet [23]	0.0073	0.0049	0.0045
Chiu [26]	0.0068	0.0055	0.0066
ACPNet [2]	0.0064	0.0048	0.0041
CGNet	0.0061	0.0049	0.0039

4.3 Ablation Experiments

To verify the importance of each module in CGNet, we conducted ablation experiments under the random distribution on the CTW dataset. Fig. 5 illustrates the impact of each module and the number of CFE module stacks on network performance. red N represents the number of CFE modules. With $N = 0$, CGNet achieves an RMSE of 0.125 m using only the feature transformation module and GFE module. When the CFE module is stacked three layers ($N = 3$), the model achieves the optimal RMSE, improving performance by 41.6% compared with that without the CFE module. These experimental results show that the shallow features extracted by the CFE module assist the GFE module to extract deeper features, thereby enhancing positioning accuracy. However, when the CFE module is stacked four layers ($N = 4$), the RMSE increases to 0.08 m. This performance degradation is due to the CFE module over-compressing the features in the deeper layers of the network, which results in a significant loss of feature information input to the GFE module. When both the feature transformation and GFE modules are removed, the model's RMSE increases to 0.083 m, representing a 12% reduction in positioning accuracy compared to when these modules are included. This confirms that the feature conversion module effectively converts features into graph structures, enabling the GFE module to extract information from these graphs.

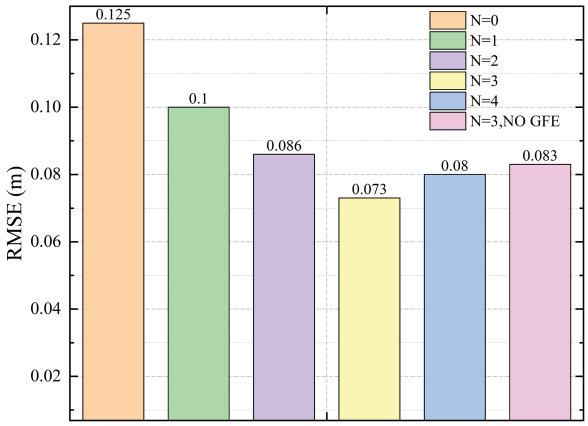


Figure 5: Ablation experiments

The selection of node distance during graph construction significantly influences model performance. Larger values of K facilitate the construction of sparser graph structures. However, excessive sparsification may result in insufficient connectivity information within the graph, thereby degrading model accuracy. As shown in Table 3, when adjusting the horizontal node distance selection, RMSE and MDE metrics exhibit an increasing trend and the positioning effect decreases when K is increased to 3 and 4. Based on this, we choose $K = 2$ for graph construction to achieve the best balance between graph sparsity and information integrity.

Table 3: Node distance performance evaluation

K	RMSE (m)	MDE (m)
1	0.076	0.065
2	0.073	0.062
3	0.077	0.069
4	0.079	0.071

As shown in Fig. 6, under the conditions of $N = 3$ and $K = 2$, the four graph construction methods exhibit significant performance differences in terms of RMSE and MDE. Experimental results demonstrate that the DAGC method achieves optimal positioning performance. Compared with the static graph construction method KNN, DAGC dynamically filters connections based on feature similarity, which effectively improves graph construction efficiency, enhances localization performance, and reduces RMSE and MDE by 8.7% and 13.8%, respectively.

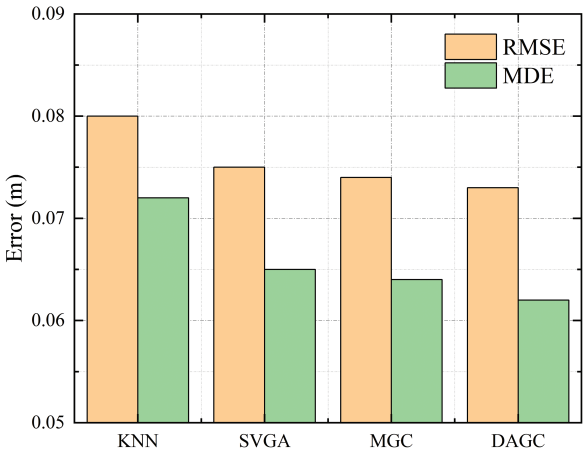


Figure 6: Comparison of graph construction methods

5 Conclusion

To address the issue that existing RVNN models cannot fully exploit the correlation between the real and imaginary parts of complex-valued CSI, we propose a lightweight indoor positioning model CGNet based on complex-valued neural networks. It combines the local feature extraction capability of complex-valued convolutions with the global relationship modeling of graph convolutions, significantly enhancing the model's feature extraction capability while maintaining its lightweight. Experimental results show that CGNet achieves a 10% improvement in positioning accuracy compared to the existing methods, while the model parameter is only 0.8 M, providing a new solution for high-precision positioning in complex indoor environments.

The lightweight structure of CGNet gives it with strong practical application potential, enabling it to adapt to diverse indoor environments while meeting the high-precision and low-resource-consumption requirements for indoor positioning imposed by IoT and smart devices. Future research directions could further explore CGNet's adaptability in dynamic environments, regarding how to maintain high-precision positioning capabilities under rapidly changing user behavior and physical environments.

Acknowledgement: Not applicable.

Funding Statement: This research was funded by Army Engineering University of PLA.

Author Contributions: The authors confirm contribution to the paper as follows: study conception and design: Le Wang and Bing Xu; data collection: En Yuan; analysis and interpretation of results: Le Wang and Peng Liu; draft manuscript preparation: Le Wang and Bing Xu. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: The data that support the findings of this study are openly available in IEEE CTW 2019 Challenge at https://data.ieeemlc.org/Ds1Detail (accessed on 26 September 2025).

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest to report regarding the present study.

References

1. Pang D, Wang G, Ho KC. Hybrid AOA-TDOA localization of a moving source by single receiver. IEEE Trans Commun. 2025;73(6):4088–104. doi:10.1109/tcomm.2024.3511932.
2. Wan R, Chen Y, Song S, Wang Z. CSI-based MIMO indoor positioning using attention-aided deep learning. IEEE Commun Lett. 2024;28(1):53–7. doi:10.1109/lcomm.2023.3335408.
3. Ruan Y, Chen L, Zhou X, Liu Z, Liu X, Guo G, et al. iPos-5G: Indoor Positioning via Commercial 5G NR CSI. IEEE Internet Things J. 2023;10(10):8718–33. doi:10.1109/jiot.2022.3232221.
4. Foliadis A, García MHC, Stirling-Gallacher RA, Thomä RS. CSI-based localization with CNNs exploiting phase information. In: IEEE Wireless Communications and Networking Conference, WCNC 2021; 2021 Mar 29–Apr 1; Nanjing, China. p. 1–6.
5. Trabelsi C, Bilaniuk O, Serdyuk D, Subramanian S, Santos JF, Mehri S, et al. Deep complex networks. arXiv:1705.09792. 2017.
6. Yu H, Liu Y, Chen M. Complex-valued neural-network-based federated learning for multiuser indoor positioning performance optimization. IEEE Internet Things J. 2024;11(21):34065–77. doi:10.1109/jiot.2024.3379872.
7. Cerar G, Svigelj A, Mohorcic M, Fortuna C, Javornik T. Improving CSI-based massive MIMO indoor positioning using convolutional neural network. In: Joint European Conference on Networks and Communications & 6G Summit, EuCNC/6G Summit 2021; 2021 Jun 8–11; Porto, Portugal; 2021. p. 276–81.
8. Zhang B, Sifaou H, Li GY. CSI-fingerprinting indoor localization via attention-augmented residual convolutional neural network. IEEE Trans Wirel Commun. 2023;22(8):5583–97. doi:10.1109/twc.2023.3235449.
9. Xiao C, Yang S, Feng Z. Complex-valued depthwise separable convolutional neural network for automatic modulation classification. IEEE Trans Instrum Meas. 2023;72:1–10. doi:10.1109/tim.2023.3298657.
10. Han K, Wang Y, Guo J, Tang Y, Wu E. Vision GNN: an image is worth graph of nodes. In: Proceedings of the 36th International Conference on Neural Information Processing Systems. Red Hook, NY, USA: Curran Associates Inc.; 2022.
11. Munir M, Avery W, Marculescu R. MobileViG: graph-based sparse attention for mobile vision applications. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops; 2023 Jun 17–24; Vancouver, BC, Canada. p. 2211–9.

12. Avery W, Munir M, Marculescu R. Scaling graph convolutions for mobile vision. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops; 2024 Jun 17–18; Seattle, WA, USA. p. 5857–65.
13. Cai Z, Wang C, Ma W, Li X, Zhou R. Lightweight automatic modulation classification based on efficient convolution and graph sparse attention in low-resource scenarios. IEEE Internet Things J. 2025;12(4):3629–38. doi:10.1109/jiot.2024.3471770.
14. Wimmer P, Mehnert J, Condurache A. Interspace pruning: using adaptive filter representations to improve training of sparse CNNs. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022; 2022 Jun 18–24; New Orleans, LA, USA. p. 12517–27.
15. Lu Z, Zhang X, Zeng R, Wang J. Better lightweight network for free: codeword mimic learning for massive MIMO CSI feedback. IEEE Commun Letters. 2023;27(5):1342–6. doi:10.1109/lcomm.2023.3258749.
16. Sun Y, Xu W, Liang L, Wang N, Li GY, You X. A lightweight deep network for efficient CSI feedback in massive MIMO systems. IEEE Wirel Commun Lett. 2021;10(8):1840–4. doi:10.1109/lwc.2021.3083331.
17. Munir M, Avery W, Rahman MM, Marculescu R. GreedyViG: dynamic axial graph construction for efficient vision GNNs. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2024; 2024 Jun 16–22; Seattle, WA, USA. p. 6118–27.
18. Li G, Müller M, Thabet AK, Ghanem B. DeepGCNs: can GCNs go as deep as CNNs? In: 2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019; 2019 Oct 27–Nov 2; Seoul, Republic of Korea. p. 9266–75.
19. Arnold M, Hoydis J, ten Brink S. Novel massive MIMO channel sounding data applied to deep learning-based indoor positioning. In: SCC 2019; 12th International ITG Conference on Systems, Communications and Coding; 2019 Feb 11–14; Rostock, Germany. p. 1–6.
20. Li C, De Bast S, Tanghe E, Pollin S, Joseph W. Toward fine-grained indoor localization based on massive MIMO-OFDM system: experiment and analysis. IEEE Sensors J. 2022;22(6):5318–28. doi:10.1109/jsen.2021.3111986.
21. Pirnat A, Bertalanic B, Cerar G, Mohorcic M, Meza M, Fortuna C. Towards sustainable deep learning for wireless fingerprinting localization. In: IEEE International Conference on Communications, ICC 2022; 2022 May 16–20; Seoul, Republic of Korea. p. 3208–13.
22. Chin W, Hsieh C, Shiung D, Jiang T. Intelligent indoor positioning based on artificial neural networks. IEEE Netw. 2020;34(6):164–70. doi:10.1109/mnet.011.2000096.
23. Cai X, Lai Q, Wang Y, Wang W, Sun Z, Yao Y. Poly kernel inception network for remote sensing detection. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2024; 2024 Jun 16–22; Seattle, WA, USA. p. 27706–16.
24. Li S, Wang Z, Liu Z, Tan C, Lin H, Wu D, et al. MogaNet: multi-order gated aggregation network. In: The Twelfth International Conference on Learning Representations, ICLR 2024; 2024 May 7–11; Vienna, Austria.
25. Li W, Meng X, Zhao Z, Liu Z, Chen C, Wang H. LoT: a transformer-based approach based on channel state information for indoor localization. IEEE Sensors J. 2023;23(22):28205–19. doi:10.1109/jsen.2023.3318835.
26. Chiu CC, Wu HY, Chen PH, Chao CE, Lim EH. 6G technology for indoor localization by deep learning with attention mechanism. Appl Sci. 2024;14(22):10395. doi:10.3390/app142210395.