GRAPH NEURAL NETWORK FOR LARGE-SCALE NETWORK LOCALIZATION

Wenzhong Yan[★]*, Di Jin*[†]*, Zhidi Lin*[★] *and Feng Yin*[★]

[★] School of Science and Engineering, The Chinese University of Hong Kong, Shenzhen, China
[†] Signal Processing Group, Technische Universitat Darmstadt, Darmstadt, Germany

ABSTRACT

Graph neural networks (GNNs) are popular to use for classifying structured data in the context of machine learning. But surprisingly, they are rarely applied to regression problems. In this work, we adopt GNN for a classic but challenging nonlinear regression problem, namely the network localization. Our main findings are in order. First, GNN is potentially the best solution to large-scale network localization in terms of accuracy, robustness and computational time. Second, proper thresholding of the communication range is essential to its superior performance. Simulation results corroborate that the proposed GNN based method outperforms all state-of-the-art benchmarks by far. Such inspiring results are theoretically justified in terms of data aggregation, non-line-of-sight (NLOS) noise removal and low-pass filtering effect, all affected by the threshold for neighbor selection. Code is available at `https://github.com/Yanzongzi/GNN-For-localization`.

Index Terms— Graph neural networks, large-scale, network localization, non-line-of-sight, thresholding.

1. INTRODUCTION

In recent years, graph neural networks (GNNs) have achieved many state-of-the-art results in various graph-related learning tasks such as node classification, link prediction and graph classification [1–4]. Comparing with the multi-layer perceptrons (MLPs), GNNs can exploit extra information from the edges. More concretely, each node aggregates information of its adjacent nodes instead of merely using its own [5, 6]. Though being effective, GNN models are often used to deal with classification tasks. While regression problems are more challenging and constitute a larger body of practical signal processing applications. In this paper, we consider a classic yet challenging nonlinear regression problem, namely network localization [7].

Network localization requires not only the measurements between agent nodes and anchor nodes, but also the measurements between the agent nodes themselves. In the past decades, a variety of canonical network localization methods have been developed, including: 1) maximum likelihood estimation based methods [7, 8]; 2) least-squares estimation based methods [9]; 3) multi-dimensional scaling based methods [10]; 4) mathematical programming based methods [11, 12] and 5) Bayesian message passing based methods [9, 13, 14].

Ignoring the effect due to non-line-of-sight (NLOS) propagation will incur severe performance degradation of the aforementioned methods. For remedy, one could perform NLOS identification for each link and then either discard the NLOS measurements or suppress them robustly in the aforementioned methods [15]. However,

accurate NLOS identification requires large-scale offline calibration and huge amount of manpower. The NLOS effect can also be dealt with from algorithmic aspect. By assuming that the NLOS noise follows a certain probability distribution, the maximum likelihood estimation based methods were developed in [16, 17]. However, model mismatch may cause severe performance degradation. In a recent work [18], network localization problem is formulated as a regularized optimization problem in which the NLOS-inducing sparsity of the ranging-bias parameters was additionally exploited. Unfortunately, all of these methods are computationally expensive for large-scale networks.

In this paper, we propose a fresh GNN based network localization method that is able to achieve all desired merits at the same time. First, it provides extremely stable and highly accurate localization accuracy despite of severe NLOS propagations. Second, it does not require laborious offline calibration nor NLOS identification. Third, it is scalable to large-scale networks at an affordable computation cost. As far as we know, this is the first time that GNN has been applied to network localization.

The remainder of this paper is organized as follows. Our problem is first formulated in Section 2. In Section 3, we introduce a fresh GNN framework for network localization. Numerical results are provided in Section 4, followed by the theoretical performance justification in Section 5. Finally, we conclude the paper in Section 6.

2. PROBLEM FORMULATION

We consider a wireless network in two-dimensional (2-D) space, and extension to the 3-D case is straightforward. We let, without loss of generality, $\mathcal{S}_a = \{1, 2, \ldots, N_l\}$ be the set of indices of the anchors, whose positions $\mathbf{p}_i = [x_i, y_i]^\top, i \in \mathcal{S}_a$ are known, and $\mathcal{S}_b = \{N_l + 1, N_l+2, \ldots, N\}$ be the set of indices of the agents, whose positions are unknown. A distance measurement made between any two nodes i and j is given by

$$x_{ij} = d(\mathbf{p}_i, \mathbf{p}_j) + n_{ij}, \tag{1}$$

where $d(\mathbf{p}_i, \mathbf{p}_j) := \|\mathbf{p}_i - \mathbf{p}_j\|$ is the Euclidean distance and n_{ij} is an additive measurement error due to line-of-sight (LOS) and NLOS propagation. A distance matrix, denoted by $\mathbf{X} \in \mathbb{R}^{N \times N}$, is constructed by stacking the distance measurements, where x_{ij} is the (ij)-th entry of $\mathbf{X}$. Notably, the distance matrix $\mathbf{X}$ is a "zero-diagonal" matrix, because $x_{ii} = 0$ for $i = 1, 2, \ldots, N$. Based on this distance matrix, our goal is to accurately locate the agents in a large-scale wireless network with satisfactory computation time.

The above signal model well suits many realistic large-scale wireless networks. For instance, in 5G network, a large number of small base stations are densely deployed in each cell; Internet of Things (IoT) network, advocating interconnection of everything,

comprises a huge number of connected smart devices and machines [19]. For large-scale networks, it is typical that only a small fraction of nodes know their locations precisely. To know all locations, otherwise, one requires either a lot of manpower to do offline calibration or to equip expensive and power-hungry GPS/BeiDou chips.

To locate a large number of agents, we propose a completely new learning paradigm, namely GNN based network localization, which is data-driven and relies on a graph-based machine learning model, to be specified in the next section.

3. NETWORK LOCALIZATION WITH GCN

Among different types of GNN models, graph convolutional networks (GCNs) constitute a representative class. In this section, we focus on formulating the network localization problem using GCNs. An undirected graph associated with a wireless network can be formally defined as $\mathcal{G} = (\mathcal{V}, \mathbf{A})$, where $\mathcal{V}$ represents the vertex set of the nodes $\{v_1, v_2, \ldots, v_N\}$, and $\mathbf{A} \in \mathbb{R}^{N \times N}$ is a symmetric adjacency matrix where a_{ij} denotes the weight of the edge between v_i and v_j. We set $a_{ij} = 1$ if there is no connection between v_i and v_j, otherwise $a_{ij} = 0$. The degree matrix $\mathbf{D} \in \mathbb{R}^{N \times N} :=$ diag$(d_1, d_2, \ldots, d_N)$ is a diagonal matrix with $d_i = \sum_{j=1}^{N} a_{ij}$.

In the original GCN, the edge a_{ij} can be regarded as a similarity coefficient between nodes i and j. In the context of network localization, two nodes are similar means that they are close to each other, i.e., $d(\mathbf{p}_i, \mathbf{p}_j)$ being small. Accordingly, we introduce a Euclidean distance threshold, denoted by T_h, to determine whether there is an edge between two nodes or not. As will be explained in Section 5, this threshold is critical to the localization performance. By thresholding, a refined adjacency matrix $\mathbf{A}_{T_h} \in \mathbb{R}^{N \times N}$ is constructed as follows:

$$[\mathbf{A}_{T_h}]_{ij} = \begin{cases} 0, & \text{if } x_{ij} > T_h \\ 1, & \text{otherwise.} \end{cases} \quad (2)$$

Consequently, the augmented adjacency matrix [1] is defined as $\tilde{\mathbf{A}}_{T_h} := \mathbf{A}_{T_h} + \mathbf{I}$, where $\mathbf{I}$ is an identity matrix, and the associated degree matrix of $\tilde{\mathbf{A}}_{T_h}$ is denoted by $\tilde{\mathbf{D}}_{T_h} \in \mathbb{R}^{N \times N}$.

Similarly, we construct a sparse distance matrix $\hat{\mathbf{X}} = \mathbf{A}_{T_h} \odot \mathbf{X}$, where $\odot$ denotes the Hadamard product. Consequently, $\hat{\mathbf{X}}$ contains only distance measurements that are smaller than or equal to T_h.

In general, each layer of GCN carries out three actions: feature propagation, linear transformation and an element-wise nonlinear activation [20]. The main difference between GCN and the standard multi-layer perceptron (MLP) [21] lies in the feature propagation, which will be clarified in the following.

In the k-th graph convolution layer, we assume D_k is the number of neurons in the k-th layer, then the input and output node representations are denoted by the matrices $\mathbf{H}^{(k-1)}$ and $\mathbf{H}^{(k)} \in \mathbb{R}^{N \times D_k}$, respectively. The initial node representations is $\mathbf{H}^{(0)} = \hat{\mathbf{X}}$. A K-layer GCN differs from a K-layer MLP in that the hidden representation of each node in GCN is averaged with its neighbors. More precisely, in GCN, the update process for all layers is obtained by performing the following matrix multiplication:

$$\bar{\mathbf{H}}^{(k)} \in \mathbb{R}^{N \times D_{k-1}} \leftarrow \hat{\mathbf{A}}_{T_h} \mathbf{H}^{(k-1)}, \quad (3)$$

where $\hat{\mathbf{A}}_{T_h} \in \mathbb{R}^{N \times N} := \tilde{\mathbf{D}}_{T_h}^{-\frac{1}{2}} \tilde{\mathbf{A}}_{T_h} \tilde{\mathbf{D}}_{T_h}^{-\frac{1}{2}}$ is the augmented normalized adjacency matrix [1] and $\bar{\mathbf{H}}^{(k)}$ is the hidden representation matrix in the k-th graph convolution layer. Intuitively, this step smoothes the hidden representations locally along the edges of the graph and ultimately encourages similar predictions among locally connected nodes.
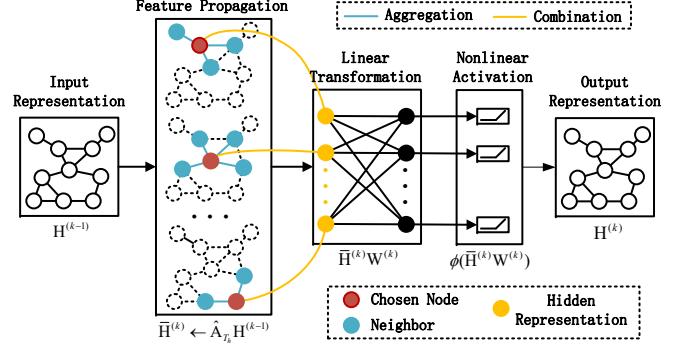

Fig. 1: Diagram of GCN updating rule for one hidden layer.

After feature propagation, the remaining two steps of GCN, i.e., linear transformation and nonlinear activation, are identical to those of the standard MLP. The k-th layer contains a layer-specific trainable weight matrix $\mathbf{W}^{(k)} \in \mathbb{R}^{D_{k-1} \times D_k}$ and a nonlinear activation function $\phi(\cdot)$, such as ReLU$(\cdot) = \max(0, \cdot)$ [22]. The representation updating rule of the k-th layer, presented in Fig. 1, is given by

$$\mathbf{H}^{(k)} \leftarrow \phi\left(\bar{\mathbf{H}}^{(k)} \mathbf{W}^{(k)}\right). \quad (4)$$

It is noteworthy that the activation function $\phi(\cdot)$ is applied to every element in the matrix.

Taking a 2-layer GCN as an example, the estimated positions, $\hat{\mathbf{R}} = [\hat{\mathbf{p}}_1, \hat{\mathbf{p}}_2, \ldots, \hat{\mathbf{p}}_N]^\top \in \mathbb{R}^{N \times 2}$, are given by

$$\hat{\mathbf{R}} = \hat{\mathbf{A}}_{T_h} \, \phi\left(\hat{\mathbf{A}}_{T_h} (\mathbf{A}_{T_h} \odot \mathbf{X}) \mathbf{W}^{(1)}\right) \mathbf{W}^{(2)}. \quad (5)$$

The weight matrices $\mathbf{W}^{(1)}$ and $\mathbf{W}^{(2)}$ can be optimized by minimizing the mean-squared-error (MSE), $\mathcal{L}(\mathbf{W}^{(1)}, \mathbf{W}^{(2)}) := \|\mathbf{R}_l - \hat{\mathbf{R}}_l\|_F^2$, where $\mathbf{R}_l = [\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_{N_l}]^\top$ and $\hat{\mathbf{R}}_l = [\hat{\mathbf{p}}_1, \hat{\mathbf{p}}_2, \ldots, \hat{\mathbf{p}}_{N_l}]^\top$ are the true anchor positions and their estimates, respectively, and $\|\cdot\|_F$ is the Frobenius norm of a matrix. This optimization problem is often solved via a gradient descent type method, such as stochastic gradient descent [23] or Adam [24].
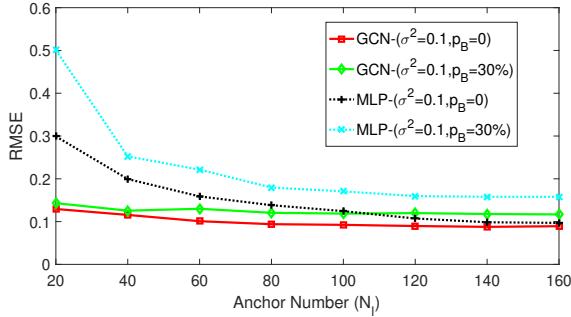
4. NUMERICAL RESULTS

In this section, the performance of the proposed GCN based method is evaluated in terms of localization accuracy, robustness against NLOS noise and computational time. For comparison purposes, we choose various well performed competitors, including an MLP based method, a neural tangent kernel (NTK) regression based method, the sparsity-inducing semi-definite programming (SDP) method [18], the expectation-conditional maximization (ECM) method [16], and the centralized least-square (LS) method [9]. Note that we choose MLP to demonstrate the performance improvement caused by adding the normalized adjacent matrix $\hat{\mathbf{A}}_{T_h}$ in each layer. Additionally, we use NTK based regression to mimic ultra-wide MLP with random initialization based on the theorem that a sufficiently wide and randomly initialized MLP trained by gradient descent is equivalent to a kernel regression predictor with the NTK [26].

Implementation details of these methods are as follows. We use a 2-layer GCN with 2000 neurons in each hidden layer. We train GCN and MLP models for a maximum number of 200 epochs (full batch size) using Adam with a learning rate of 0.01. We initialize the weights using the routine described in [27] and normalize the input feature vectors along rows. Dropout rate [28] is set to 0.5 for all

Table 1: The averaged loss (RMSE) of all methods under different noise conditions for N_l=50.

Methods\ Noise (σ^2, p_B)	$(0.04, 0\%)$	$(0.1, 10\%)$	$(0.25, 10\%)$	$(0.25, 30\%)$	$(0.5, 50\%)$
GCN	**0.1038**	**0.1128**	**0.1006**	**0.1302**	**0.1755**
GCN_{1000}	0.0874	0.0856	0.0998	0.0981	0.1404
MLP	0.1865	0.1769	0.2305	0.2623	0.3358
NTK [25]	0.4307	0.5155	0.5270	0.6154	0.9578
SDP [18]	0.1171	0.2599	0.4891	0.4641	0.9294
ECM [16]	0.1610	0.1857	0.3298	0.3824	0.8011
LS [9]	0.2270	0.2675	0.3884	0.4187	0.7992
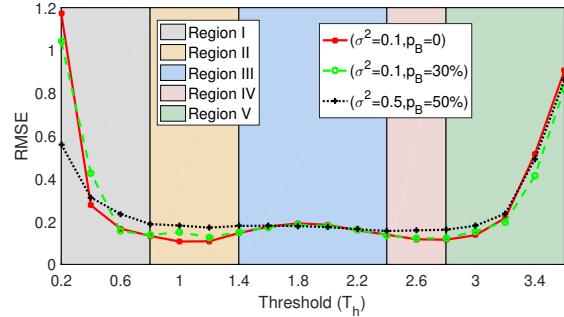
Fig. 2: The averaged loss (RMSE) versus the number of anchors under different noise conditions.

Fig. 3: The averaged loss (RMSE) versus threshold under different noise conditions in GCN model. $N_l = 50$.

layers. The settings of NTK remain the same as described in [25]. The regularization parameter in SDP is set to $\lambda = 0.05$. For both the ECM and LS methods, the initial positions are randomly generated in the square area. All simulations are performed on a server computer equipped with 48 Inter Xeon E5-2650 2.2GHz CPUs and 8 NVIDIA TITAN Xp 12GB GPUs. In all experiments, we set the threshold $T_h = 1.2$ for GCN, MLP and NTK, and set $T_h = 0.6$ for other methods, which leads to similar averaged localization accuracy but requires much less computational time than using $T_h = 1.2$. Fairness in comparison is carefully maintained.

Details of the simulated localization scenarios are given below. We consider a 5m×5m unit square area. Each network consists of 500 randomly generated nodes in total. The number of anchors, N_l, varies from 20 to 160 for investigating its impact, and the rest are agents. The measurement error n_{ij} is generated according to $n_{ij} = n_{ij}^L + b_{ij} n_{ij}^N$. Here, the LOS noise n_{ij}^L is generated from a zero-mean Gaussian distribution, i.e., $n_{ij}^L \sim \mathcal{N}(0, \sigma^2)$, while the positive NLOS bias n_{ij}^N is generated from the uniform distribution, $n_{ij}^N \sim \mathcal{U}[0, 10]$ and b_{ij} generated from the Bernoulli distribution $\mathcal{B}(p_B)$ with p_B being the NLOS occurrence probability.

First, we assess the localization accuracy of all methods under different noise conditions. Here, the localization accuracy is measured in terms of the averaged test root-mean-squared-error (RMSE), $\mathcal{L}_R := \|\mathbf{R}_u - \hat{\mathbf{R}}_u\|_F$, where $\mathbf{R}_u = [\mathbf{p}_{N_l+1}, \mathbf{p}_{N_l+2}, \ldots, \mathbf{p}_N]^\top$ and $\hat{\mathbf{R}}_u = [\hat{\mathbf{p}}_{N_l+1}, \hat{\mathbf{p}}_{N_l+2}, \ldots, \hat{\mathbf{p}}_N]^\top$. The results are summarized in Table 1. It is shown that among all considered methods, GCN provides the highest localization accuracy in almost all cases. In particular, when the NLOS probability, p_B, is high, GCN outperforms all competitors by far. Moreover, we test the localization performance of GCN for large networks with $N = 1000$, denoted by GCN_{1000} in Table 1. The results show that GCN performs even better with slightly increased computational time, as shown in Table 2. If we further increase N, the GCN based method can maintain its performance, but the other methods (SDP, ECM and LS) will all degrade severely in terms of localization accuracy and computational time.

Next, we focus on two data-driven methods, GCN and MLP, which perform relatively well in Table 1. The localization accuracy is investigated by varying N_l from 20 to 160 with a stepsize of 20 under two different noise conditions. The results are depicted in Fig. 2. There are two main observations. First, GCN attains the lowest RMSE consistently for all N_l. Compared with MLP, the improvement in localization accuracy is particularly remarkable for GCN with small N_l. This result indicates that GCN can better exploit the distance information than MLP. When N_l increases, both GCN and MLP tend to approach a performance lower bound. Second, GCN performs similarly under both noise conditions, while MLP shows a clear performance degradation when p_B increases. This observation indicates that GCN is very robust against NLOS. Lastly, we want to mentioned that a fine-tuned MLP is often superior to NTK which corresponds to random initialized MLP, which performs surprisingly close to other benchmark methods as shown in Table 1.

In the third simulation, we focus on investigating the influence of the threshold, T_h, on the localization performance. Figure 3 depicts the RMSE of GCN versus the threshold, T_h, in three different noise scenarios. It is interesting to see that the RMSE curves show similar trend as the threshold changes. We characterize the localization performance obtained by using different thresholds. In **Region I** ($T_h \in [0.2, 0.8]$), the RMSE is very large at the beginning and drops rapidly as T_h increases. The reason for such bad performance at the beginning is that when T_h is too small there will be no sufficient edges in the graph, incurring isolated nodes. In **Regions II $\sim$ IV** ($T_h \in (0.8, 2.8]$), GCN shows stable performance. A closer inspection shows that the RMSE is relatively lower in **Region II**, rises slightly in **Region III** and decreases in **Region IV** to the same lowest level as in **Region II**. This observation can be explained as follows. When $T_h \in (0.8, 1.4]$, the good performance of GCN is due to the NLOS noise truncation effect of T_h, which will be explained in Section 5.1. For $T_h \in (2.4, 2.8]$, the adverse effect of large NLOS noise is compensated by the increased number of neighbors for each node. Lastly, the rapid increase of RMSE in **Region V** can be explained by the effect of extremely large NLOS noise and over-smoothing, which will be explained in Section 5.1 as well.

Table 2: A comparison of different methods in terms of computational time (in second) at $(\sigma^2 = 0.1, p_B = 30\%)$ and $N_l = 50$.

GCN	GCN$_{1000}$	GCN$_{10000}$	MLP	NTK	LS	ECM	SDP
3.24	5.82	707.38	**2.05**	2.33	32.47	82.85	1587

Another important requirement for real-world applications is fast response. Table 2 shows the practical computational time of different methods. It is shown that GCN, MLP and NTK are more computationally efficient than the other methods. Besides, the computational time of GCN$_{1000}$ only slightly increases when we double the number of nodes in the network. Notably, GCN can process very large network, for instance $N = 10000$, in an affordable time, while the LS, ECM and SDP are all disabled in this case. All above results indicate that the proposed GCN based method is a prospective solution to large-scale network localization.

5. PERFORMANCE REASONING

In this section, we aim to dig out the reasons behind the remarkable performance of the newly proposed GCN based method, corroborated by the results shown in Section 4. We pinpoint two major factors: one is the threshold T_h and the other is the augmented normalized adjacency matrix $\hat{\mathbf{A}}_{T_h}$. In the following, we analyze the two factors separately, although T_h determines $\hat{\mathbf{A}}_{T_h}$.

5.1. Effects of Thresholding

Thresholding plays two major roles: 1) truncating large noise and 2) avoiding over-smoothing.
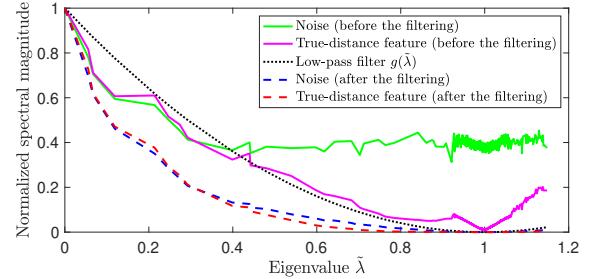
Noise truncation. The operation, $\hat{\mathbf{X}} = \mathbf{A}_{T_h} \odot \mathbf{X}$, in Section 3 implies that for each $\hat{x}_{ij} \neq 0$, $d(\mathbf{p}_i, \mathbf{p}_j) + n_{ij} \leq T_h$ holds. Equivalently speaking, for each non-zero element in $\hat{\mathbf{X}}$, we have $n_{ij} \leq T_h - d(\mathbf{p}_i, \mathbf{p}_j)$, indicating that only noise in a small limited range will be included in $\hat{\mathbf{X}}$. Specifically, due to the fact that n_{ij} with large value is usually caused by positive NLOS bias n_{ij}^N, each element x_{ij}, associated either with a large true distance or with a large noise, is neglected when the threshold T_h is set small. In other words, we retain the measurement if the two nodes are adjacent and only affected by a small or moderate measurement noise.

Avoiding over-smoothing. When the threshold is large enough, the corresponding graph will become fully connected and the adjacency matrix will be a matrix of all ones. In this case, according to Eq. (3), all entries of the hidden representation matrix $\bar{\mathbf{H}}^{(k)}$ are identical, meaning that the obtained hidden representation completely loses its distance information. Consequently, the predicted positions of all nodes will tend to converge to the same point. This phenomenon is known as over-smoothing. As an illustration, **Region V** in Fig. 3 confirms that GCN will suffer from over-smoothing when the threshold is set too large. Thus, we need to choose a proper threshold to prevent such an adversarial behavior.

5.2. Effects of $\hat{\mathbf{A}}_{T_h}$

To understand the superior localization performance of GCN compared with MLP, we analyze the effects of the augmented normalized adjacency matrix, $\hat{\mathbf{A}}_{T_h}$, from both spatial and spectral perspectives.

Aggregation and combination. To understand the spatial effect of $\hat{\mathbf{A}}_{T_h}$, we decompose $\hat{\mathbf{A}}_{T_h}$, cf. Eq. (3), into two parts:


Fig. 4: Spectral components of different signals in dataset ($\sigma^2 = 0.1, p_B = 0$).

$$\bar{\mathbf{h}}_i^{(k)} = \underbrace{\frac{1}{d_i+1}\mathbf{h}_i^{(k-1)}}_{\text{Own information}} + \underbrace{\sum_{j=1}^{N} \frac{a_{ij}}{\sqrt{(d_i+1)(d_j+1)}}\mathbf{h}_j^{(k-1)}}_{\text{Aggregated information}}, \quad (6)$$

where $\bar{\mathbf{h}}_i^{(k)}$ and $\mathbf{h}_i^{(k)}$ are the i-th row vectors of hidden representation matrix, $\bar{\mathbf{H}}^{(k)}$, and the input representation matrix, $\mathbf{H}^{(k)}$, in the k-th layer, respectively. Specifically, Eq. (6) contains two operations: aggregation and combination. Aggregation corresponds to the second term of Eq. (6), in which the neighboring features are captured by following the given graph structure. Then, the target node's own information is combined with the aggregated information.

Comparing with the training procedure of MLP, which solely uses the features of the labeled nodes (referred to as anchors here), GCN is a semi-supervised method in which the hidden representation of each labeled node is averaged for a carefully tailored local neighborhood including itself. Equivalently speaking, GCN trains a model by exploiting features of both labeled and unlabeled nodes, leading to superior localization performance.

Low-pass filtering. From the spectral perspective, the eigenvalues of the augmented normalized Laplacian $\mathbf{L}_{T_h} = \mathbf{I} - \hat{\mathbf{A}}_{T_h}$, denoted by $\tilde{\lambda}$, can be regarded as the "frequency" components [29, 30]. Multiplying K augmented normalized adjacency matrices $\hat{\mathbf{A}}_{T_h}^K$ in graph convolution layers is equivalent to passing a spectral "low-pass" filter $g(\tilde{\lambda}_i) := (1 - \tilde{\lambda}_i)^K$, where $\tilde{\lambda}_i, i = 1, 2, \dots$ [20]. Figure 4 depicts the "frequency" components of the LOS noise and the true distance matrix before and after the filtering process. It can be seen that almost all information of the true distance matrix (before the filtering) is concentrated in the "low frequency" band, while both "low frequency" and "high frequency" components are present in the LOS noise before the filtering. Thus, $\hat{\mathbf{A}}_{T_h}$, acting as a "low-pass" filter, can partially remove the "high frequency" component of the LOS noise. This explains the improved localization performance of GCNs from the spectral perspective.

6. CONCLUSION

In this paper, we have proposed a GCN based data-driven method for robust large-scale network localization in mixed LOS/NLOS environments. Numerical results have shown that the proposed method is able to achieve substantial improvements in terms of localization accuracy, robustness and computational time, in comparison with both MLP and various state-of-the-art benchmarks. Moreover, our detailed analyses found that thresholding the neighboring features is crucial to attaining superb localization performance. The proposed data-driven paradigm is believed to drive more efficient and robust methods for network localization and related ones in the future.

5253

7. REFERENCES

[1] T. N. Kipf and M. Welling, "Semi-Supervised Classification with Graph Convolutional Networks," in *International Conference on Learning Representations (ICLR)*, Toulon, France, April 2017.

[2] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive Representation Learning on Large Graphs," in *Advances in neural information processing systems (NIPS)*, 2017, pp. 1024–1034.

[3] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph Attention Networks," 2017.

[4] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How Powerful are Graph Neural Networks," 2019.

[5] A. Ortega, P. Frossard, J. Kovačević, J. M. Moura, and P. Vandergheynst, "Graph signal processing: Overview, challenges, and applications," *Proceedings of the IEEE*, vol. 106, no. 5, pp. 808–828, 2018.

[6] F. Gama, A. G. Marques, G. Leus, and A. Ribeiro, "Convolutional neural network architectures for signals supported on graphs," *IEEE Transactions on Signal Processing*, vol. 67, no. 4, pp. 1034–1049, 2018.

[7] N. Patwari, A. O. Hero, M. Perkins, N. S. Correal, and R. J. O'Dea, "Relative Location Estimation in Wireless Sensor Networks," *IEEE Transactions on Signal Processing*, vol. 51, no. 8, pp. 2137–2148, Aug 2003.

[8] A. Simonetto and G. Leus, "Distributed Maximum Likelihood Sensor Network Localization," *IEEE Transactions on Signal Processing*, vol. 62, no. 6, pp. 1424–1437, 2014.

[9] H. Wymeersch, J. Lien, and M. Z. Win, "Cooperative Localization in Wireless Networks," *Proceedings of the IEEE*, vol. 97, no. 2, pp. 427–450, 2009.

[10] J. A. Costa, N. Patwari, and A. O. Hero, "Distributed Weighted-Multidimensional Scaling for Node Localization in Sensor Networks," *ACM Transactions on Sensor Networks*, vol. 2, no. 1, pp. 39–64, 2006.

[11] P. Biswas, T. Lian, T. Wang, and Y. Ye, "Semidefinite Programming Based Algorithms for Sensor Network Localization," *ACM Transactions on Sensor Networks*, vol. 2, no. 2, pp. 188–220, May 2006.

[12] P. Tseng, "Second-Order Cone Programming Relaxation of Sensor Network Localization," *SIAM Journal on Optimization*, vol. 18, no. 1, pp. 156–185, 2007.

[13] A. Ihler, J. W. Fisher, R. L. Moses, and A. S. Willsky, "Non-parametric Belief Propagation for Self-Localization of Sensor Networks," in *IEEE Journal on Selected Areas in Communications*, 2005.

[14] D. Jin, F. Yin, C. Fritsche, F. Gustafsson, and A. M. Zoubir, "Bayesian cooperative localization using received signal strength with unknown path loss exponent: Message passing approaches," *IEEE Transactions on Signal Processing*, vol. 68, pp. 1120–1135, 2020.

[15] S. Marano, W. M. Gifford, H. Wymeersch, and M. Z. Win, "NLOS Identification and Mitigation for Localization Based on UWB Experimental Data," *IEEE Journal on Selected Areas in Communications*, vol. 28, no. 7, pp. 1026–1035, Sep. 2010.

[16] F. Yin, C. Fritsche, D. Jin, F. Gustafsson, and A. M. Zoubir, "Cooperative Localization in WSNs Using Gaussian Mixture Modeling: Distributed ECM Algorithms," *IEEE Transactions on Signal Processing*, vol. 63, no. 6, pp. 1448–1463, 2015.

[17] H. Chen, G. Wang, Z. Wang, H. C. So, and H. V. Poor, "Non-Line-of-Sight Node Localization Based on Semi-Definite Programming in Wireless Sensor Networks," *IEEE Transactions on Wireless Communications*, vol. 11, no. 1, pp. 108–116, 2012.

[18] D. Jin, F. Yin, M. Fauß, M. Muma, and A. M. Zoubir, "Exploiting Sparsity for Robust Sensor Network Localization in Mixed LOS/NLOS Environments," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 5915–5915.

[19] F. Yin, Z. Lin, Q. Kong, Y. Xu, D. Li, S. Theodoridis, and S. R. Cui, "Fedloc: Federated learning framework for data-driven cooperative localization and location data processing," *IEEE Open Journal of Signal Processing*, vol. 1, pp. 187–215, 2020.

[20] F. Wu, T. Zhang, A. H. d. Souza Jr, C. Fifty, T. Yu, and K. Q. Weinberger, "Simplifying Graph Convolutional Networks," in *International Conference on Learning Representations (ICLR)*, 2019.

[21] D. Svozil, V. Kvasnicka, and J. Pospichal, "Introduction to Multi-layer Feed-forward Neural Networks," *Chemometrics and Intelligent Laboratory Systems*, vol. 39, no. 1, pp. 43–62, 1997.

[22] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for Activation Functions," in *International Conference on Learning Representations (ICLR)*, 2018.

[23] L. Bottou, "Large-Scale Machine Learning with Stochastic Gradient Descent," in *Proceedings of 19th International Conference on Computational Statistics*. 2010, pp. 177–186, Springer.

[24] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in *International Conference on Learning Representations (ICLR)*, 2014.

[25] S. Arora, S. S. Du, Z. Li, R. Salakhutdinov, R. Wang, and D. Yu, "Harnessing the Power of Infinitely Wide Deep Nets on Small-Data Tasks," in *International Conference on Learning Representations (ICLR)*, 2020.

[26] S. Arora, S. S. Du, W. Hu, Z. Li, R. R. Salakhutdinov, and R. Wang, "On Exact Computation with an Infinitely Wide Neural Net," in *Advances in Neural Information Processing Systems (NIPS)*, 2019, pp. 8141–8150.

[27] X. Glorot and Y. Bengio, "Understanding the Difficulty of Training Deep Feedforward Neural Networks," in *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*, 2010, pp. 249–256.

[28] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[29] A. Sandryhaila and J. M. Moura, "Discrete signal processing on graphs: Frequency analysis," *IEEE Transactions on Signal Processing*, vol. 62, no. 12, pp. 3042–3054, 2014.

[30] F. Gama, E. Isufi, G. Leus, and A. Ribeiro, "Graphs, convolutions, and neural networks: From graph filters to graph neural networks," *IEEE Signal Processing Magazine*, vol. 37, no. 6, pp. 128–138, 2020.