Centrality-aware gossiping for distributed learning in wireless sensor networks

J.S. Mertens, L. Galluccio, G. Morabito
CNIT Research Unit at Department of Electrical, Electronics and Informatics Engineering (DIEEI)
University of Catania
Viale Andrea Doria, 6 - 95125 Catania, Italy

***Abstract*—In recent years, federated learning and other distributed machine learning solutions have emerged to enable collaborative synthesis of machine learning models in wireless sensor networks. However, these approaches require the exchange of large volumes of data to reach convergence, which is costly in terms of communication and computing resources. Gossiping, which naturally fits with the multihop communication paradigm featured in most wireless sensor networks application scenarios, can reduce the amount of exchanged data significantly. In this paper we investigate how the exploitation of topological information about the wireless sensor network can accelerate the convergence and also improve efficiency in terms of network resource consumption. More specifically, we introduce a gossiping learning protocol which exploits the centrality information to reduce the communication rounds needed and, thus, the amount of data exchanged. We performed a large number of experiments by considering different centrality measures and observed that exploitation of centrality information makes the convergence faster when compared to the case in which such information is not used.**

***Index Terms*—Wireless sensor networks, distributed training, centrality, gossiping, federated learning.**

I. Introduction

Advances in machine learning (ML) techniques and emergence of privacy concerns in networking have led to the increasing interest for the so called concept of *Federated Learning* (FL). FL is a distributed machine learning solution that enables the training of a ML model across multiple peripheral edge devices, while keeping the data locally in the devices themselves. Thus, FL computes the ML model in a privacy preserving way [1] without disclosing data through the network. However, in the case of *wireless sensor networks* (WSN), FL solutions require large number of transmissions of model parameters and result in high consumption of energy and communication resources at those nodes located closer to the node which performs aggregation of the individual models. This is indeed a well drawback denoted as funneling effect [2].

On the other hand, gossiping is a mechanism in which nodes achieve a shared goal by exchanging local information with their neighbouring nodes. In this way energy and communication resources consumption can be reduced, thus improving network lifetime [3], [4] as well. The collaborative learning of a model through gossiping in a network is referred to as gossip learning [5], [6].

Distributed learning through gossiping solves the problems of FL in WSNs and has proved to be efficient. Similarly to FL, in gossip learning, data collected by sensors remains local at the devices. However, differently from FL, there is not a unique node or central component responsible for performing the aggregation of all models. Rather, models are exchanged and aggregated locally. However, gossip learning still requires the consumption of a considerable amount of energy and communication resources before the distributed learning processes across all nodes converge. In this context, the policy applied by nodes to determine when to send their model is a critical factor for learning, since it impacts on the convergence rate.

Therefore, in this paper we propose a gossiping method which exploits the centrality information to achieve the convergence rapidly, thus reducing the consumption of energy and communication resources. This paper is organized as follows. In section II, we recall the well known centrality metrics used in WSNs. In Section III, we discuss the Centrality-aware Gossiping for Distributed learning (CGL) protocol in detail and its operations. In Section IV, we present the experimental setup considered for our experiments, followed by the numerical results. Finally in Section V, conclusions are drawn.

II. Centrality in WSN

Centrality characterizes the topological *importance* of a node in a network and its relationships with its neighbouring nodes. In this section, we discuss some of the applications of centrality metrics in WSN scenarios [7], [8].

The most commonly used centrality metrics are: *Betweennes*, *Eigen*, *Pagerank*, *degree, centrality* and *closeness centrality* measures. The *Betweeness centrality* measure assesses the number of times a node lies on the shortest path between other pairs of nodes. The *Eigen centrality* assesses a node impact by considering the importance of its one-hop neighbours. More specifically, it measures a node influence based on the number of links it has to the other nodes in the network. Additionally, it also considers how well connected those other nodes are, and how many connections they have in the network. The *Pagerank* centrality is a variant of the Eigen centrality. Similarly to the latter, it assigns a score to a node based on its connections and its connections of connections. However, Pagerank also considers the direction

This work was partially supported by SAFE-DEMON and MUR under contract $PON03PE\_00214\_2$ "DELIAS", as well as by University of Catania, under contract PIACERI.

ISBN 978-3-903176-48-5

of the link and the connection weight [9]. *Degree centrality* is the simplest centrality measure that assesses the number of links held by each node in the network. More specifically, it denotes the one-hop connections each node has with its neighbouring nodes. The *Closeness centrality* of a node is the geodesic distance between itself and all other nodes in the network. More specifically, it assesses how short the shortest paths are between the node and all other nodes.

Centrality information is typically used in Cluster Head selection problems in WSN scenarios for enhancing network lifetime by optimizing cluster head location. This is crucial in scenarios that exploit clustering to reduce resource consumption [10], [11].

Energy efficient routing is a critical issue in WSN scenarios since it is necessary for prolonging the network lifetime and Centrality metrics have been studied and applied for designing efficient routing algorithms [12], [13] In our work, we study and design a centrality-aware gossiping protocol for distributed learning, called Centrality-aware Gossiping for Distributed Learning (CGL), which enables achieving quick convergence, thus resulting in significantly increased network lifetime.

III. Centrality-aware Gossiping mechanism

In this section, we present the Centrality-aware Gossiping for Distributed Learning (CGL) protocol designed and tested in this work. More specifically, in Section III-A, we discuss the design objectives of CGL. Then, in Section III-B, we illustrate the protocol operations in detail.

A. Design objectives

The core objective in distributed learning solutions, such as federated learning and gossip-based learning, is to solve a distributed optimization problem in the form [14]:

$$\min_{\mathbf{w}} F(\mathbf{w}) \tag{1}$$

where

$$F(\mathbf{w}) = \sum_{k=1}^{K} p_k F_k(\mathbf{w}) \tag{2}$$

In eq. (2) $F(\mathbf{w})$ is the objective function, $\mathbf{w}$ is the vector containing the model parameters, K is the number of network nodes, called *learners*[1], p_k is the weight of the k-th learner such that $\sum_{k=1}^{K} p_k$=1 and $F_k(\mathbf{w})$ is the local loss function for the $k-th$ node when the model is $\mathbf{w}$.

Accordingly, the target of CGL is to support the nodes in collaboratively learning the ML model, $\mathbf{w}$, by exploiting the gossiping mechanism in a WSN. In such a scenario, a given node will transmit its model parameters to its one-hop neighbouring nodes that will update their own model with the received parameters and train it further.

For instance if j is the node that has transmitted its model parameters, $\mathbf{w}_j$, its generic neighbor k will update its model as

$$\mathbf{w}_k = \alpha \cdot \mathbf{w}_j + (1-\alpha) \cdot \mathbf{w}_k \tag{3}$$

[1]In this paper we will use the terms "*node*" and "*learner*" interchangeably.

where α is a weight parameter the setting of which will be discussed later. Note that node k performs the operation in eq. (3) if the model loss obtained by using $\mathbf{w}_j$ is lower than the loss obtained using its current model, i.e., $F_k(\mathbf{w_j}) < F_k(\mathbf{w_k})$. We call this sequence of actions "*communication round*" or "*iteration*", in short.

Another characteristic of CGL is that, each node in the network should exploit its local information only, i.e., the centrality information and the data generated by the node itself. Thus, the operations of CGL should be asynchronous and uncoordinated. Due to the energy limitations in WSNs, the operations of CGL should be as effective as possible, with the objective to reduce the global loss function (see eq. (2)) by requiring minimum number of model transmissions.

To achieve the above goals, each node, k, of the network knows and stores its centrality metric and at each iteration schedules the transmission of its model parameters after a time interval, τ_k, with average that is proportional to ratio

$$E\{\tau_k\} = F_k(\mathbf{w}_k)/(B_k \cdot CE_k) \tag{4}$$

where CE_K is the centrality of the node and B_k is called *boost factor* and is described in the following.

Furthermore, the communication and model training load must be distributed between all network nodes as fairly as possible. This is necessary to have good fairness among nodes and avoid overloading of a limited number of nodes. To achieve this goal, CGL exploits the *boost factor*. A node k increments its *boost factor* whenever it receives the model parameters sent by one of its neighbors, say j. Instead, the *boost factor* is reset to one when the node broadcasts its model parameters in the CGL mechanism. Given that the boost factor is at the denominator of eq. (4), the value of τ_k will be higher for nodes that did not transmit their model parameters in the recent past and smaller for nodes that transmitted their model parameters recently.

B. Protocol Details

In this section, we provide more details regarding the operations of CGL and how it works in a toy exemplary WSN scenario. Then, in the next Section IV, we will consider a realistic scenario.

Let us consider a network consisting of four nodes deployed according to a linear topology as shown in Figure 1.

Algorithm 1 reports the pseudo-code for the protocol operations executed by a generic node k. In the pseudocode CE_k represents the centrality measure of node k.

At setup, any node initializes the boost factor $B_k = 1$. Furthermore, the model is trained using the local data X_k. The training operation is executed for a given number of epochs, at each node separately. Epochs denote the number of times the machine learning algorithm will work through the entire dataset during the training.

Also at setup, each CGL node is initialized by calculating its centrality measure $CE(k)$. To this purpose, observe that several distributed algorithms exist that allow such calculation efficiently (see, for example, [15]–[18]).

At the end of the initialization phase which is illustrated in lines 1-5 of Algorithm 1, the CGL node k will have a local

model with parameters $\mathbf{w}_k$ and an estimate of its current loss $F_k(\mathbf{w}_k)$. Using these initial parameters the node sets a timeout at time $t_k^{(\text{next})} = t + T \cdot F_k(\mathbf{w}_k)/(B_k \cdot CE_k)$, where t represents the current time and T is a constant value which is set in such a way that protocol operations are sparse in time[2].

For example in Figure 1, the loss values for the four nodes at the end of the initialization phase are $F_1(\mathbf{w}_1) = 226.73$, $F_2(\mathbf{w}_2) = 92.11$, $F_3(\mathbf{w}_3) = 76.34$, and $F_4(\mathbf{w}_4) = 70.42$, respectively.

Let us consider the Degree centrality for this example. Therefore the centrality values for the four nodes are $CE_1 = 1$, $CE_2 = 2$, $CE_3 = 2$, and $CE_4 = 1$. Consequently, the four nodes will set their timeouts at times $t_1^{(\text{next})} = 226.73 \cdot T$, $t_2^{(\text{next})} = 46.05 \cdot T$, $t_3^{(\text{next})} = 38.17 \cdot T$, and $t_4^{(\text{next})} = 70.42 \cdot T$, respectively.

Note that Figure 1 was obtained assuming that the setup time is $t = 0$ for all nodes.

When the initialization phase is completed, CGL executes the regular operations, sketched in lines 7-27 of the Algorithm 1. More specifically,

1) when the timeout $t_k^{(\text{next})}$ elapses, the node executes the operations sketched in lines 9-15 of Algorithm 1;
2) when a given node receives the model parameters, $\mathbf{w}_j$, from the neighboring node j, it executes the operations sketched in lines 17-25 of Algorithm 1, including re-training the model for a given number of epochs.

As reported in lines 9-15 of Algorithm 1, when the timeout elapses, the node will broadcast its model parameters $\mathbf{w}_j$ to its neighbouring nodes. In the example shown in Figure 1, node 3 has the lowest value of the timeout parameter, i.e. $t_3^{(\text{next})} = 38.17 \cdot T$.

Therefore, as reported in line 11, it will broadcast a message containing its model parameters $\mathbf{w}_3$. Note that the time at which a node disseminates its model parameters depends on the timeout value. As an example, the first dissemination performed by node 3 occurs at time $t = 38.17 \cdot T$. The disseminated message will be received by nodes 2 and 4 which will execute the operations sketched in lines 17-25 of Algorithm 1.

Note that the nodes 2 and 4 first increase the boost factor by one, evaluate the loss they would achieve by exploiting the model $\mathbf{w}_3$ (i.e. $F_2(\mathbf{w}_3)$ and $F_4(\mathbf{w}_3)$). If the estimated value of loss is higher than the current one, no other actions will be executed and the node will wait for a new event. Instead, if the estimated value of loss, $F_k(\mathbf{w}_j)$, is lower than or equal to the current one, $F_k(\mathbf{w}_k)$, the node executes the operations in lines 21-24. Thus, node 2 and 4 update their model parameters $\mathbf{w}_2$ and $\mathbf{w}_4$ as $\mathbf{w}_2 = \alpha \cdot \mathbf{w}_3 + (1-\alpha) \cdot \mathbf{w}_2$ and $\mathbf{w}_4 = \alpha \cdot \mathbf{w}_3 + (1-\alpha) \cdot \mathbf{w}_4$ and, then, train the model using their local data and evaluate new loss values. Finally, they update the $t_k^{(\text{next})}$ value.

In the example shown in Figure 1, the resulting value of $F_2(\mathbf{w}_2)$ at time $t = 38.17 \cdot T$ is 86.12. Since $CE_2 = 2$ and $B_2 = 2$, the timeout is set at time $t_2^{(\text{next})} = 59.7 \cdot T$. Similarly, the resulting value of $F_4(\mathbf{w}_4)$ at the same time is 54.11. Since $CE_4 = 1$ and $B_4 = 2$, the timeout is set at time $t_4^{(\text{next})} = 65.22 \cdot T$.

In the next round, i.e., at time $t = 59.7 \cdot T$, node 2 broadcasts its model parameters to nodes 1 and 3.

Then it will be the turn of node 3 that will transmit its new model parameters, $\mathbf{w}_3$.

In just three transmissions of model parameters, the values of the loss in the four nodes become $F_1(\mathbf{w}_1) = 90.84$, $F_2(\mathbf{w}_2) = 79.74$, $F_3(\mathbf{w}_3) = 44.33$, and $F_4(\mathbf{w}_4) = 36.44$. Such values are much lower than the initial ones and, as a result, the average loss, which was equal to (226.73+92.11+76.34+70.42)/4=116.4 at time 0, rapidly decreases to (90.84+79.74+44.33+36.44)/4=62.81.

[2]In our experiment, we have set T equal to a constant proportional to the average time required to train the model local data.

Algorithm 1 CGL protocol

1: */* Protocol initialization */*
2: Initialize B_k=1
3: Calculate Centrality metrics CE_k
4: Initialize $\mathbf{w}_k$=train_model($\mathbb{X}_k$, RND)
5: $t_k^{(\text{next})} = t + T \cdot F_k(\mathbf{w}_k)/(B_k \cdot CE_k)$
6: */* Regular operations */*
7: **while** `Protocol initialization` **do**
8: Wait for `Event`
9: **if** `Event.Type`==time-out **then**
10: */* Node k will send its model parameters $\mathbf{w}_k$ */*
11: `Broadcast`($\mathbf{w}_k, F_k(\mathbf{w}_k)$)
12: `Wait for the updated loss values`
13: $B_k = 1$
14: $t_k^{(\text{next})} = t + T \cdot F_k(\mathbf{w}_k)/(B_k \cdot CE_k)$
15: **end if**
16: **if** `Event.Type` == received $\mathbf{w}_j$ **then**
17: */* Node k is receiving the model parameters $\mathbf{w}_j$ from neighbor node j */*
18: $B_k = B_k + 1$
19: `Evaluate` $F_k(\mathbf{w}_j)$
20: **if** $F_k(\mathbf{w}_j) \leq F_k(\mathbf{w}_k)$ **then**
21: $\mathbf{w}_k = \alpha \cdot \mathbf{w}_j + (1-\alpha) \cdot \mathbf{w}_k$
22: $\mathbf{w}_k$ = train_model($\mathbb{X}_k, \mathbf{w}_k$)
23: `Evaluate` $F_k(\mathbf{w}_k)$
24: $t_k^{(\text{next})} = t + T \cdot F_k(\mathbf{w}_k)/(B_k \cdot CE_k)$
25: **end if**
26: **end if**
27: **end while**

IV. Experimental Results

In this section, we apply CGL protocol to a specific WSN scenario. In Section IV-A, we describe the scenario with the dataset considered and the machine learning algorithms used in the experiments. In Section IV-B, we present and discuss the numerical results obtained from the simulation experiments.

A. WSN Scenario

For our experiments, we considered the data-set containing the values of temperature, humidity, and concentrations of several pollutants (i.e., PM1, PM2.5, PM10) measured by 56

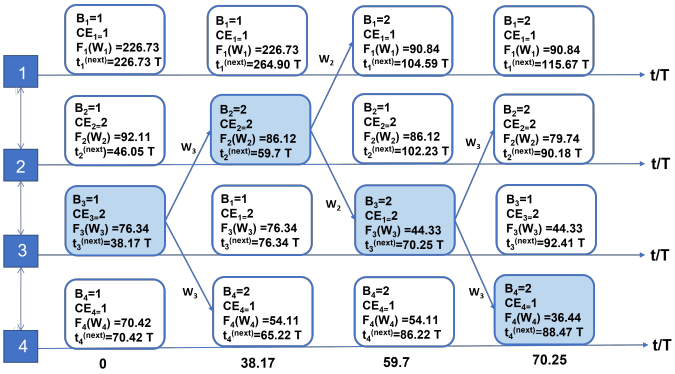

Fig. 1: CGL protocol in action

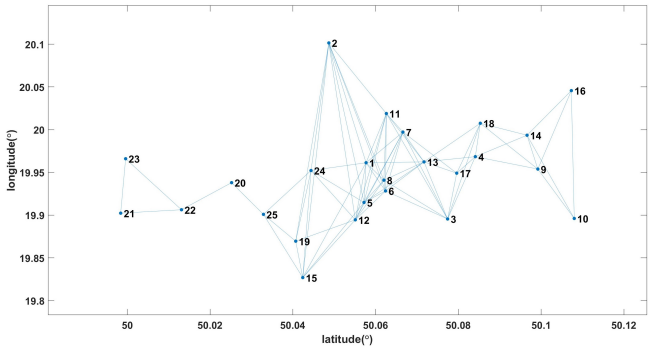

Fig. 2: Wireless network scenario used for experiments.

low cost sensors deployed in the city of Krakow (Poland) in 2017[3] Since, some of the sensors do not have complete data, among these, we selected 25 sensors which provide a complete dataset. The topology of the sensor network, determined by considering the radio coverage of each node equal to 4 km, is shown in Figure 2.

The sensors collect one value for each parameter every hour, therefore the data-set for one month (i.e. 30 days) considered for each of the 25 sensors consists of $n_k = 24 \cdot 30 = 720$ entries of 7 values, with $k = 1, 2, ..., 25$. These 7 values represent the day, time, temperature, humidity, and PM1, PM2.5, and PM10 parameters measured in the time period July 1-30, 2017. Note that the day and time values are not included for training the model, and thus the 7 parameters reduce to only 5.

Accordingly, the u-th entry in the dataset of the k-th sensor, denoted as $\mathbf{X}_{k,u}$, is:

$$\mathbf{X}_{k,u} = (X_{k,u}^{(0)}, X_{k,u}^{(1)}, , ..., X_{k,u}^{(4)}) \tag{5}$$

CGL can be applied whatever is the ML approach utilized. For our experiments we consider two cases with different machine learning approaches. In Case 1, we assume that each sensor node executes an autoencoder [19] which represents a traditional methodology for dimension reduction, denoising, and anomaly detection [20]. In Case 2, we assume each sensor node executes a LSTM (Long Short-Term Memory) neural network for time series prediction.

For the centrality metrics, we considered betweenness, closeness, pagerank, degree and eigen centrality metrics.

1) Case 1: Autoencoders: Autoencoders are an unsupervised neural networks utilized for learning efficient data representation. More specifically, a bottleneck is imposed in the neural network that forces a compressed representation of the input data.

Concerning the ML approach, in our experiments we have employed the simplest type of autoencoder, the - so called - *Vanilla autoencoder* which consists of one hidden layer, only. Accordingly, in our experiments, the autoencoder consists of a three layer neural network, thus having one input, one hidden, and one output layer.

Note that the input size of the auto-encoder used in the experiments is 5, i.e., the values of temperature, humidity, PM1, PM2.5 and PM10. The intermediate size is 3 which is also known as code or compressed dimension.

2) LSTM Network: The LSTM neural network is a kind of *recurrent neural network* (RNN) approach that is trained through the back-propogation mechanism over time. LSTM neural networks are applied for forecasting tasks [21].

In our case study, we exploit the LSTM to predict the next hour PM1 values using the regression approach. More specifically, we use the PM1 values from the dataset to create two columns of data: the first column containing the PM1 values (t) and the second column containing the next hour PM1 values (t+1), to be predicted. Thus the LSTM model trains on the two columns to map a function for predicting the next hour PM1 values.

B. Numerical Results

In this section, we report the numerical results obtained by applying the CGL protocol in the scenarios described in previous Section IV-A. We conducted a simulation campaign to observe the average loss of the sensor nodes obtained using different centrality metrics in both the cases, i.e. autoencoder and LSTM.

More specifically, 10 simulations were executed in both the cases for different values of α (i.e. α =0.5, 0.7, 0.9 and 1), different number of *epochs* (i.e., 10, 15 and 20) and different centrality metrics. Epochs denote the number of times the machine learning model will work through the entire dataset during the training.

Previously, in order to select which centrality measure performs better, in Figure 3, we report the average loss obtained with respect to the overall amount of data transmitted by network nodes in the autoencoders case when α is 0.7 and the number of epochs is set to 15. We observe that the average loss in case of the Betweenness centrality converges faster when compared to the other centrality metrics. We also observe that the cases with Degree centrality and Page rank centrality had relatively slower convergence rates. Note that each iteration in the CGL protocol corresponds to one packet transmission.

Similarly in Figure 4 we report the average loss obtained with respect to the amount of data trasmitted in the LSTM

[3]https://www.kaggle.com/datascienceairly/air-quality-data-from-extensive-network-of-sensors

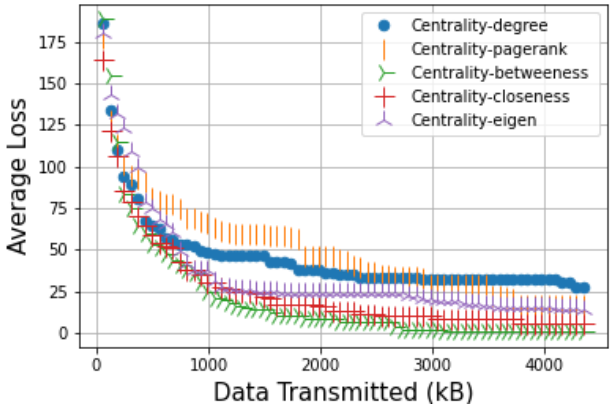

Fig. 3: Average Loss vs. Data Transmitted for different centrality measures (Autoencoders)

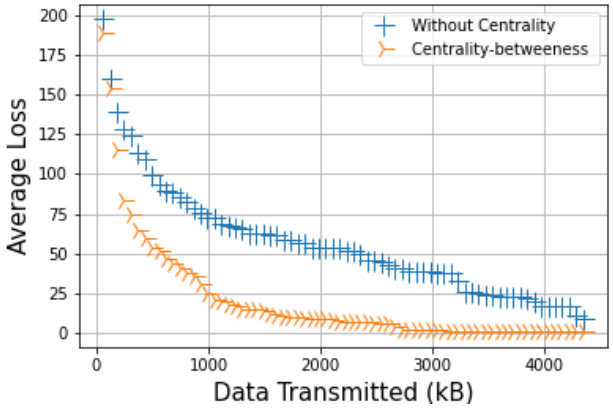

Fig. 5: Average Loss vs. Data Transmitted (Autoencoders)

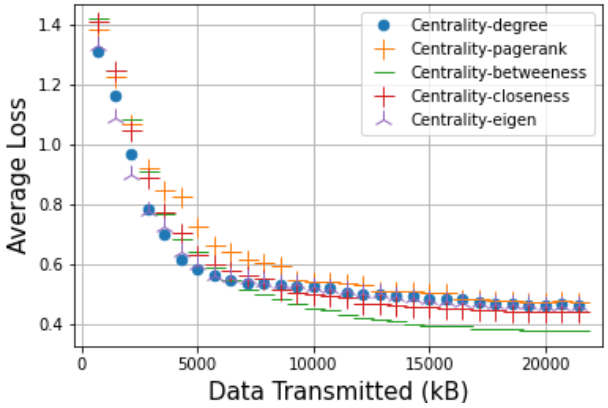

Fig. 4: Average Loss vs. Data Transmitted for different centrality measures (LSTM-Prediction)

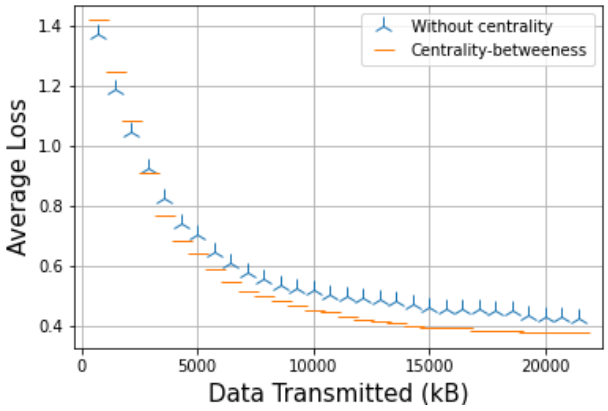

Fig. 6: Average Loss vs. Data Transmitted (LSTM-Prediction)

prediction case when α is 0.7 and the number of epochs is 15. Similarly to the autoencoder case, we observe faster convergence when considering the Betweenness centrality metrics and relatively slower convergence with Degree and Pagerank centrality metrics.

Since considering Betweenness centrality gives better performance in both the cases, in the following we focus on such centrality metric to make a comparison with the case in which distributed learning is based on a gossiping scheme which does not consider the centrality measures of nodes.

In Figure 5, we report the average loss with respect to the amount of data transmitted in the autoencoder case. We observe a significant improvement in performance in the betweeness centrality case when compared to the case without any centrality metrics.

We also did a similar comparison in Case 2 and we noticed a similar behavior as shown in Figure 6. This assesses that the Betweenness centrality plays an important role in reducing the number of communication rounds required to collaboratively learn a fitting model in a WSN.

Furthermore, in Figure 7 we illustrate the number of times a sensor node has been trained using CGL in Case 1, estimated until the loss converges and stabilizes. In the figure, nodes are sorted out in decreasing order of number of training cycles (i.e., not based on their IDs). We observed that the majority of nodes were trained approximately 20 times or more; on the other hand, few sets of nodes were trained less than 10 times. This implies that the training event depends on how influential a node is in the network.

We also observe similar results in the case of LSTM-prediction (Case 2), as shown in Figure 8. In this case, the majority of nodes were trained 8 times or more, whereas a set of 7 nodes were trained 5 times or less. Similarly to case 1, nodes which were less trained, are those exhibiting lower Betweeness centrality. From both these cases, we notice that the centrality metric of a node influences the training event in both the cases.

Thus, numerical analysis assesses how the centrality metric can significantly impact gossiping and enhance the collaborative learning approach of machine learning models in a WSN scenario.

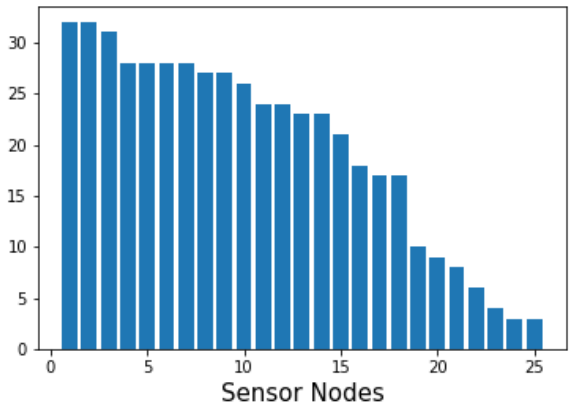

Fig. 7: Number of times each node has been trained (Case 1: Autoencoder)

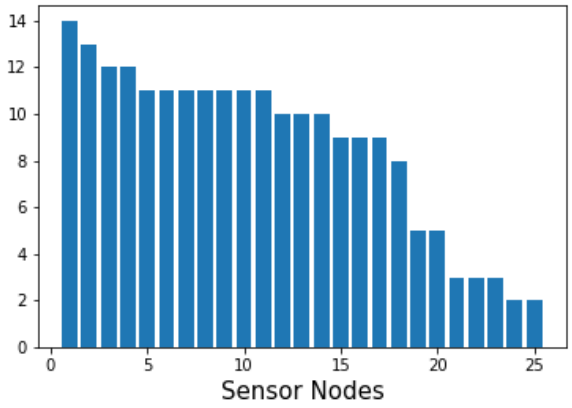

Fig. 8: Number of times each node has been trained (Case 2: LSTM-Prediction)

V. Conclusions

Distributed learning has emerged as an efficient methodology in enabling collaborative computing of a machine learning model in a wireless sensor network. However, this approach implies numerous communication rounds for the convergence which is costly in terms of communication and computing resources.

Accordingly, in this paper we have introduced the CGL mechanism which exploits the centrality of the nodes for gossiping in distributed learning in WSN. The operations of CGL protocol have been described in details and performance evaluated in a specific WSN scenario; the impact of different centrality measures in gossiping has been illustrated and compared. Simulation results assess the effectiveness of exploiting the centrality measure of nodes to increase the efficiency of distributed learning in WSNs.

References

[1] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," 2017.

[2] J. Mena, M. Gerla, and V. Kalogeraki, "Mitigate funnel effect in sensor networks with multi-interface relay nodes," in *2012 IEEE 8th International Conference on Distributed Computing in Sensor Systems*, pp. 216–223, 2012.

[3] A. G. Dimakis, S. Kar, J. M. F. Moura, M. G. Rabbat, and A. Scaglione, "Gossip algorithms for distributed signal processing," *Proceedings of the IEEE*, vol. 98, no. 11, pp. 1847–1864, 2010.

[4] D. Shah., "Gossip algorithms," *Foundations and Trends in Networking*, vol. 3, no. 1, 2009.

[5] I. Hegedüs, G. Danner, and M. Jelasity, "Gossip learning as a decentralized alternative to federated learning," in *DAIS*, 2019.

[6] J. Mertens, L. Galluccio, and G. Morabito, "Federated learning through model gossiping in wireless sensor networks," in *2021 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom)*, pp. 1–6, 2021.

[7] L. C. Freeman, "Centrality in social networks conceptual clarification," *Social Networks*, vol. 1, no. 3, pp. 215–239, 1978.

[8] A. Jain and B. Reddy, "Node centrality in wireless sensor networks: Importance, applications and advances," in *2013 3rd IEEE International Advance Computing Conference (IACC)*, pp. 127–131, 2013.

[9] P. Zhang, T. Wang, and J. Yan, "Pagerank centrality and algorithms for weighted, directed networks with applications to world input-output tables," 2021.

[10] I. Gupta, D. Riordan, and S. Sampalli, "Cluster-head election using fuzzy logic for wireless sensor networks," in *3rd Annual Communication Networks and Services Research Conference (CNSR'05)*, pp. 255–260, 2005.

[11] V. Aditya, S. Dhuli, P. L. Sashrika, K. K. Shivani, and T. Jayanth, "Closeness centrality based cluster head selection algorithm for large scale wsns," in *2020 12th International Conference on Computational Intelligence and Communication Networks (CICN)*, pp. 107–111, 2020.

[12] E. M. R. Oliveira, H. S. Ramos, and A. A. F. Loureiro, "Centrality-based routing for wireless sensor networks," in *2010 IFIP Wireless Days*, pp. 1–5, 2010.

[13] A. Jain, "Betweenness centrality based connectivity aware routing algorithm for prolonging network lifetime in wireless sensor networks," *Wirel. Netw.*, vol. 22, p. 1605–1624, jul 2016.

[14] B. Luo, X. Li, S. Wang, J. Huang, and L. Tassiulas, "Cost-effective federated learning design," 2020.

[15] A.-M. Kermarrec, E. Le Merrer, B. Sericola, and G. Trédan, "Second order centrality: Distributed assessment of nodes criticity in complex networks," *Computer Communications*, vol. 34, no. 5, pp. 619–628, 2011.

[16] K. Wehmuth and A. Ziviani, "Daccer: Distributed assessment of the closeness centrality ranking in complex networks," *Computer Networks*, vol. 57, no. 13, pp. 2536–2548, 2013.

[17] K. You, R. Tempo, and L. Qiu, "Distributed algorithms for computation of centrality measures in complex networks," *IEEE Transactions on Automatic Control*, vol. 62, no. 5, pp. 2080–2094, 2016.

[18] R. Kumar Behera, S. Kumar Rath, S. Misra, R. Damaševičius, and R. Maskeliūnas, "Distributed centrality analysis of social network data using mapreduce," *Algorithms*, vol. 12, no. 8, p. 161, 2019.

[19] W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu, and F. E. Alsaadi, "A survey of deep neural network architectures and their applications," *Neurocomputing*, vol. 234, pp. 11–26, 2017.

[20] e. J. K. Chow, "Anomaly detection of defects on concrete structures with the convolutional autoencoder," *Elsevier Advanced Engineering informatics*, vol. 45, 2020.

[21] Y. Duan, Y. L.V., and F.-Y. Wang, "Travel time prediction with lstm neural network," in *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, pp. 1053–1058, 2016.