Data-Driven and Privacy-Preserving Cooperation in Decentralized Learning

Data-Driven and Privacy-Preserving Cooperation in Decentralized Learning

Francesco Malandrino
CNR-IEIIT and CNIT

Carlos Barroso Fernandez
Universidad Carlos III de Madrid

Carlos J. Bernardos Cano
Universidad Carlos III de Madrid

Carla Fabiana Chiasserini
Politecnico di Torino, CNR-IEIIT,
Chalmers University of Technology

Antonio De La Oliva
Universidad Carlos III de Madrid

Mahyar Onsori
Politecnico di Torino

***Abstract*—Decentralized learning scenarios offer the opportunity of a *flexible* cooperation between learning nodes; in other words, each node may cooperate with an arbitrary subset of its peers. In such scenarios, we tackle the problem of choosing the nodes that cooperate towards the training of a machine learning model, hence, tweaking the *cooperation graph* connecting the nodes themselves. We propose and evaluate a *data-driven* approach to the problem, by proposing three metrics to choose the edges to activate in the cooperation graph, and an efficient iterative algorithm exploiting them. Through our performance evaluation, which leverages state-of-the-art datasets and neural network architectures, we find that privacy-preserving metrics accounting for the difference between local datasets are very effective in identifying the best edges to activate to improve the efficiency of model training without hurting performance.**

I. Introduction

The capabilities of machine learning (ML) models keep growing at a very fast pace and have became highly beneficial and faintly alarming at the same time. In particular, a relevant aspect is the *training* of such models, which requires ever-increasing amounts of data and computational resources, e.g., CPU and GPU cores. As a consequence, the training of large-scale ML models may take advantage of multiple *learning nodes*, sharing both their resources and their local datasets. The cooperation between nodes typically follows one of two main paradigms, namely, distributed learning or decentralized learning.

In distributed learning, all nodes train one model, each using its local data, and nodes are assisted by a coordinator (a.k.a. aggregator). At each iteration, each node sends its local model to the aggregator, which combines them (e.g., by averaging) and sends the global model it obtains back to the learning nodes. Federated learning [1] and variants thereof [2], [3] all follow such distributed training paradigm. Conversely, *decentralized learning* [4], [5], exemplified in Fig. 1, adopts altogether different cooperation and communication strategies. There is no centralized coordinator, rather nodes directly communicate with each other. Furthermore, nodes only exchange the gradients for the model that is trained, and, most importantly, the cooperation between nodes is *flexible*, i.e., nodes are not required to exchange such gradients with all other nodes. Instead, nodes are free to decide the neighbors

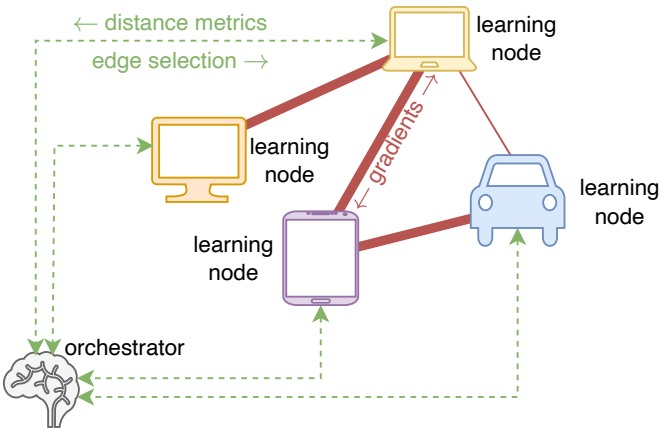

Fig. 1. A decentralized learning scenario and conceptual representation of the proposed solution. Four *learning nodes* with heterogeneous capabilities and information have the option of cooperating to perform a DNN training task. Cooperation takes the form of exchanging gradients; nodes that can communicate are connected by solid, red edges, which may (thick edges) or may not (thin ones) be used for exchanging gradients (i.e., be active). In the scenario we envision, nodes are assisted by an *orchestrator*: nodes send to the orchestrator information about their dataset (e.g., distance metrics) and the orchestrator determines the most cost-effective cooperation graph. It then sends back to the nodes indications on which of their neighbors they should cooperate with. Such communication is represented by green, dashed edges in the figure.

to cooperate with, building a *cooperation graph* like that formed by the red edges in Fig. 1. Thanks to its higher flexibility and the fact that it does not require a coordinator of the training process, decentralized learning is best suited to fully distributed, heterogeneous scenarios, where a coordinator cannot be easily identified (or, it is not convenient to do so) and nodes deemed to cooperate have different capabilities.

As also discussed in Section II, choosing the pairs of nodes that cooperate, i.e., deciding the cooperation graph, is critical for the overall learning performance. Accordingly, we envision that learning nodes are assisted by a learning *orchestrator* (depicted as the brain in Fig. 1). Importantly, as per the decentralized learning paradigm, learning nodes do not send models to the orchestrator; rather, they share the information needed to identify the most appropriate edges to add to the

cooperation graph. The orchestrator combines such information, determines the best cooperation graph, and sends it to the learning nodes, as shown by the green lines in Fig. 1. In general, the orchestrator will seek to minimize the *cost* of the overall learning process, subject to learning quality (e.g., accuracy) constraints: adding more edges will, in general, improve the learning quality [4], [5], while also increasing the incurred cost [4].

In this work, we tackle the problem that the orchestrator needs to solve to optimally select the edges of the cooperation graph, accounting for two complementary viewpoints: (i) *how* to make the decisions, and (ii) which *information* to use to make such decisions. Concerning the former, we devise an efficient, iterative algorithm yielding provably near-optimal solutions. For the latter, we propose and evaluate three different *metrics* to assess how useful each edge can be, featuring different trade-offs between the decision quality and the extent to which they preserve the privacy of local datasets.

The remainder of this paper is organized as follows. We begin from a motivating example in Section II, showing how significant the impact of the cooperation graph on the learning quality is. Then, in Section III we present our system model and an optimization formulation of the problem solved by the orchestrator. In view of the problem complexity, we present our solution concept, called PickEdge, in Section IV, and evaluate its performance in Section V. After discussing related work in Section VI, we conclude the paper in Section VII.

II. Motivating Examples: Edges Matter

Our PickEdge approach envisions adding a new entity to the traditional decentralized learning scenario – the orchestrator in Fig. 1 –, providing it with information, and having it make some fairly complex decisions about learning organization. This is of course worthwhile if good decisions about the cooperation graph can be made that significantly improve the learning quality. Importantly, the orchestrator *does not* aggregate models trained locally and has no part in the training process, i.e., the training procedure still fully meets the principles of decentralized learning.

As a motivating example, let us consider a simple decentralized learning scenario, based upon widely-used, publicly-available datasets and DNN architectures. Specifically, we consider nine learning nodes that have to perform an activity classification task over the HAR dataset [6] using the feed-forward DNN from [7]. Each node has a local dataset whose size varies between 400 and 600 samples. Furthermore, the number of samples of each class in each local dataset are assigned randomly, which results in a non-uniform data distribution.

In this scenario, we consider a total of 9,450 different cooperation graphs, and track, for each of them, (i) the number of edges therein and (ii) the accuracy it yields. The results are summarized in Fig. 2, where each marker corresponds to a cooperation graph, and the marker's locations along the x- and y-axes correspond (respectively) to the number of edges in the graph and the resulting accuracy. Moreover, the purple area in the plot corresponds to the values between the 10th and 90th percentiles of accuracy.

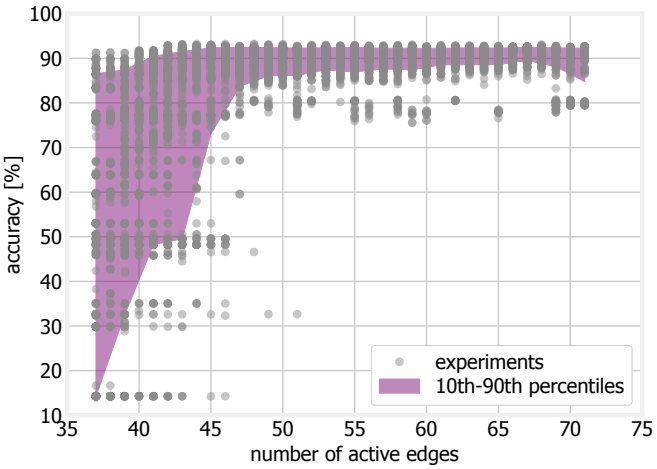

Fig. 2. Why the cooperation graph matters. We consider an activity classification task based upon [6] and over 9,000 cooperation graphs. Markers in the plot correspond to cooperation graphs, and their locations along the x- and y-axes correspond (respectively) to the number of edges in the graph and the resulting accuracy. The purple area encloses the 10th and 90th percentiles of accuracy.

A first observation we can make from the plot is that, as expected, more cooperation (hence, more edges) almost always results in better performance. It is more interesting to remark what happens for less connected cooperation graphs, e.g., with around 40 edges. In these cases, the resulting accuracy can be as low as 10% – the same as randomly guessing – but also as high as 90%, matching that yielded by cooperation graphs with twice the number of edges. Achieving one accuracy value or the other solely depends upon the quality of our decisions about the cooperation graph.

An alternative way of seeing the high-level purpose of our work and the PickEdge framework is looking at the purple area, corresponding to the solution space we can explore. Moving left results in cheaper solutions; moving down results in lower-quality ones. PickEdge aims at moving as much as possible towards the left, while avoiding going down; ideally, we would move across the top edge of the purple area. Notice how the edge of such purple area is relatively straight, i.e., *there are* solutions that are both cheap and effective, hence, it is worthwhile looking for them.

III. System Model and Problem Formulation

The main element of our system model, exemplified in Fig. 3, is the set of learning nodes $\mathcal{N}=\{n\}$, representing the nodes that can participate in the learning process. We further know a set of edges $\mathcal{E}=\{(n_1,n_2)\}\subseteq\mathcal{N}^2$, representing the pairs of nodes that *could* cooperate with each other. For each edge, we are given a per-epoch cost $c(n_1,n_2)$ representing, e.g., the energy consumed if nodes n_1 and n_2 cooperate.

Our main decision is whether to activate each edge, expressed through binary variables $y(n_1,n_2)\in\{0,1\}$. Additionally, we have to decide the number K of epochs to run. Given

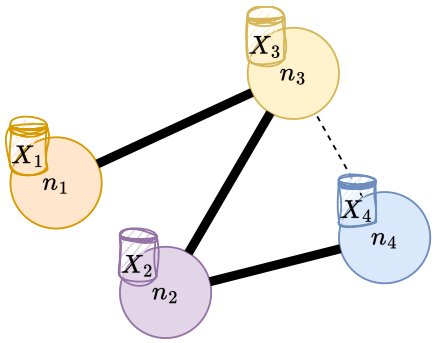

Fig. 3. An example scenario with four nodes in $\mathcal{N}$ (represented by circles) and four edges in $\mathcal{E}$ (represented by lines). To each node corresponds a local dataset X_n, represented by a cylinder. The style of lines denotes whether or not the corresponding edge is active; as an example, $y(n_1, n_3)$=1, $y(n_3, n_4)$=0, while $(n_1, n_2) \notin \mathcal{E}$.

the vector $\mathbf{y}$ of all y-variables and K, the achieved loss is given by $\ell(K, \mathbf{y})$; we impose that such a loss value must be equal to or smaller than a target value $\ell^{\max}$.

Our high-level goal is to minimize the total training cost, subject to the fact that the loss target is met:

$$\min_{K,y} \quad K \sum_{(n_1,n_2)\in\mathcal{E}} y(n_1, n_2) c(n_1, n_2), \tag{1}$$

$$\text{s.t. } \ell(K, \mathbf{y}) \leq \ell^{\max}. \tag{2}$$

There are, however, two main issues with the problem as it is stated above. Firstly, there is no closed-form expression for the loss $\ell(K, \mathbf{y})$, and even estimating the impact of decisions over the loss is not straightforward. Secondly, the problem as a whole is NP-hard; specifically, it is an integer non-linear programming problem, or INLP, where the non-linearity comes from the definition of $\ell(K, \mathbf{y})$. It follows that it is impractical to solve it directly through, e.g., a solver. We deal with both issues by proposing:

- three possible metrics to estimate the value of activating a given edge;
- an iterative algorithm, based on hill-climbing, which makes near-optimal decisions.

IV. The PickEdge Solution

Our solution concept, called PickEdge, features two main components. First, we replace the loss $\ell(K, \mathbf{y})$ with an estimate of the value of enabling each edge, as detailed in Section IV-A. Then, we use such value estimates in our iterative algorithm, which is presented in Section IV-B, prove that it makes near-optimal decisions.

A. Assigning values to edges

Given an inactive edge $(n_1, n_2) \in \mathcal{E}$ (i.e., such that $y(n_1, n_2)$=0), the current values of the y-variables $\mathbf{y} \in \{0, 1\}^{|\mathcal{E}|}$, and the number K of epochs to run, the value $v(n_1, n_2, K, \mathbf{y})$ associated with edge (n_1, n_2) is defined as the loss improvement (i.e., decrease) achieved by enabling the edge itself, i.e.,

$$v(n_1, n_2, K, \mathbf{y}) = \ell(K, \mathbf{y}) - \ell(K, \mathbf{y}'), \tag{3}$$

where $\mathbf{y}'$ is a copy of $\mathbf{y}$, with the value corresponding to (n_1, n_2) set to 1. Here, we consider that, enabling additional edges does not hurt performance, hence, $v(n_1, n_2, K, \mathbf{y})$ is always non-negative.

Exact edge values are impractical to compute, given the stochastic nature of the DNN training process. Furthermore, having those values depending on both the number K of epochs to run and the currently-selected edges $\mathbf{y}$ is very cumbersome, and significantly adds to the problem complexity. Accordingly, in the following we present three *metrics* to assess edge values, leveraging different types of information.

The spectral metric $\widetilde{\mathbf{v}}_\gamma(n_1, n_2, \mathbf{y})$. The spectral gap $\gamma(\mathbf{y})$ of a graph is the difference between the moduli of the two largest eigenvalues of its incidence matrix. It has been observed in the literature [5, Eq. (5)] that such a quantity has an influence on the learning quality; specifically, a lower spectral gap results in a higher loss. Furthermore, adding an edge to a graph increases the spectral gap[1]. Accordingly, the spectral metric $\widetilde{v}_\gamma(n_1, n_2, \mathbf{y})$ assigns to each edge (n_1, n_2) a value corresponding to how much enabling that edge increases the spectral gap, i.e.,

$$\widetilde{v}_\gamma(n_1, n_2, \mathbf{y}) = \gamma(\mathbf{y}') - \gamma(\mathbf{y}). \tag{4}$$

The spectral metric is the easiest to compute, as it requires no information whatsoever about the nodes or their datasets. On the negative side, not accounting for such information may lead to sub-optimal performance.

The dataset distance metric $\widetilde{\mathbf{v}}_\mathbf{D}(n_1, n_2, \mathbf{y})$ It has been observed in the literature [8] that including nodes with overly-different local dataset can hurt the overall learning performance. Accordingly, calling X_n the local dataset of node n, the data distance estimate of the value of edge (n_1, n_2) is the opposite of the distance between their datasets:

$$\widetilde{v}_\mathrm{D}(n_1, n_2) = -\|X_{n_1} - X_{n_2}\|. \tag{5}$$

Notice how, unlike (4), the definition in (5) does not depend upon the current decisions $\mathbf{y}$. It follows that the dataset distance metric can be computed offline, and does not change as decisions evolve.

The V-distance metric $\widetilde{\mathbf{v}}_\mathbf{V}(n_1, n_2, \mathbf{y})$ A major disadvantage of the data distance metric (5) is that it requires to share the local datasets X_n. This is problematic for two main reasons: it creates additional overhead, and it may jeopardize privacy. To avoid such issues, we *decompose* the local datasets using singular value decomposition (SVD) decomposition, and represent them as:

$$X_n = U_n \Sigma_n V_n^\intercal, \tag{6}$$

where Σ_n is a diagonal matrix, and the matrices V_n and U_n can be interpreted, respectively, as a description of the dataset as a whole (akin to a base) and a representation of the individual samples therein. Accordingly, the distance (5) between the datasets can be well approximated through the distance

[1] A fully-connected topology is associated with a spectral gap of 1; all other topologies have gap values between 0 and 1.

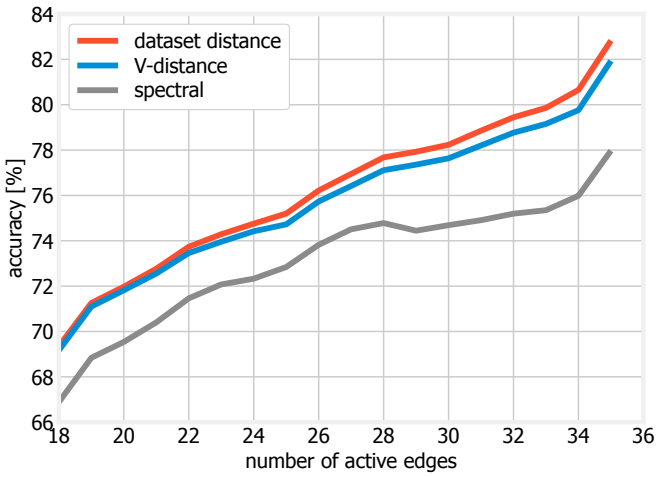

Fig. 4. Accuracy achieved when different metrics are employed by PickEdge, as a function of the number of edges selected to be included in the cooperation graph.

Algorithm 1 The PickEdge algorithm

Require: $K, \ell^{\max}$

1: **for all** $(n_1, n_2) \in \mathcal{E}$ **do**
2: $\quad y(n_1, n_2) \leftarrow 0$
3: **while** $\ell(K, \mathbf{y}) \geq \ell^{\max} \wedge \min \mathbf{y} = 0$ **do**
4: $\quad (n_1^\star, n_2^\star) \leftarrow \arg\max_{\substack{(n_1,n_2)\in\mathcal{E} \\ : y(n_1,n_2)=0}} \frac{v(n_1,n_2,K,\mathbf{y})}{c(n_1,n_2)}$
5: $\quad y(n_1^\star, n_2^\star) \leftarrow 1$
return $\mathbf{y}$

between the corresponding V-matrices, giving the following metric:

$$\widetilde{v}_{\mathrm{V}}(n_1, n_2) = -\|V_{n_1} - V_{n_2}\|. \tag{7}$$

Importantly, *V-matrices are much smaller than the original datasets (i.e., the X-matrices), and can be exchanged between nodes with negligible overhead and without jeopardizing privacy*.

B. The PickEdge algorithm

The PickEdge algorithm, summarized in Alg. 1, follows a straightforward greedy approach. We start with no edges enabled (Line 2); then, until the learning quality is attained (Line 3), we choose a new edge to activate. Specifically, we identify the inactive edge that maximizes the ratio between its value – however estimated, as per Section IV-A – and the cost (Line 4), and set the corresponding y-variable to 1 (Line 5). Notice that, for ease of presentation, Alg. 1 assumes K to be given; if that is not the case, multiple instances of the algorithm could be ran with different values of K.

In spite of its simplicity, Alg. 1 has two interesting properties. First, its worst-case computational complexity is polynomial – indeed, linear in the number of edges in $\mathcal{E}$. Second, for all the definitions of edge value presented in Section IV-A, it yields a performance (i.e., the total cost in (1)), that is within a constant factor from the optimum.

More formally, we can prove that:

Property 1. *The worst-case computational complexity of the PickEdge algorithm (in Alg. 1) is linear in the number of edges $|\mathcal{E}|$.*

Proof: The proof comes from inspection of Alg. 1 where the only loop starts in Line 3. Every time the loop runs one value of $\mathbf{y}$ is set to 1, and when all of its values are set to 1, the check in Line 3 cannot succeed. Since the length of $\mathbf{y}$ is $|\mathcal{E}|$, this is also the maximum number of times the loop can run, hence, it represents the worst-case complexity of the algorithm as a whole. ■

Concerning the competitive ratio, the following holds:

Property 2. *For all of the metrics value definitions presented in Section IV-A, the the PickEdge algorithm (in Alg. 1) has a constant competitive ratio of $1-\frac{1}{\mathrm{e}}$.*

Proof: The proof comes from [9, Theorem 4.7], showing that greedy algorithms in the form of Alg. 1 have a competitive ratio of $1-\frac{1}{\mathrm{e}}$, if the value function is submodular.

Concerning the three functions defined in Section IV-A, (5) and (7) are linear (i.e., do not depend upon current decisions $\mathbf{y}$), hence, they are also submodular. As for (4), we leverage the experiments in [10], [11], showing that the increase in the spectral gap achieved by adding one edge decreases with the spectral gap itself – the very definition of submodularity. ■

It is however important to remark that the performance of Alg. 1 can only be as good as the value estimates provided to it. In other words, if the estimates are wrong, then so will the decisions returned by the algorithm.

V. Performance Evaluation

For our performance evaluation, we consider the same reference scenario as in Section II, with nine nodes performing a decentralized learning task. The first aspect we are interested in is the performance achieved by PickEdge when different metrics are used, quantified through the classification accuracy.

The results are summarized in Fig. 4. As one might expect, a larger number of edges (hence, more cooperation) is always associated with a better accuracy. Indeed, as also shown in [5], cooperation always improves accuracy – unless very specific distributions of data and computation times are considered. It is more interesting to observe the difference between the metrics presented in Section IV-A: the spectral metric (4) yields the lowest accuracy, with dataset distance (5) and V-distance (7) providing higher, and very similar, performance.

The reason for the relatively poor performance of the spectral metric can be seen from (4) itself. This metric does not account for anything other than the topology of the cooperation graph, hence, employing that metric may result in not exploiting properly nodes with more, and/or more useful, data. As far as the dataset distance (5) is concerned, it consistently yields the best performance – which is linked to the fact that the dataset distance metric captures the most

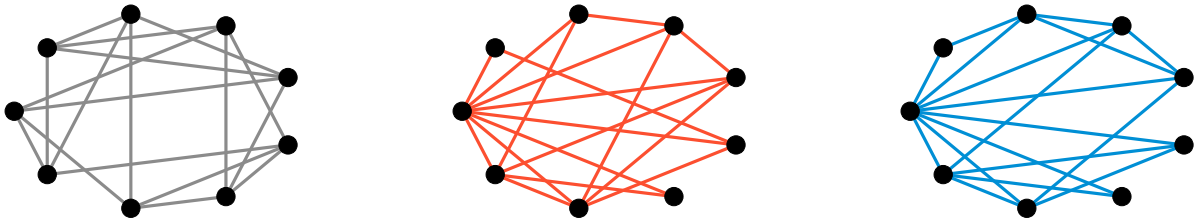

Fig. 5. Cooperation graphs yielded by PickEdge when the target number of edges is 18 and the used metric is spectral (left), dataset distance (center), V-distance (right).

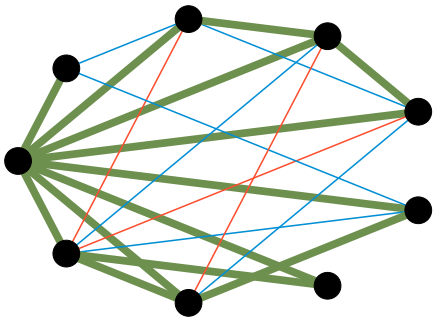

Fig. 6. 18-edge cooperation graphs: edges activated under both the dataset distance and V-distance metrics (green); under dataset distance only (red); and, under V-distance only (blue).

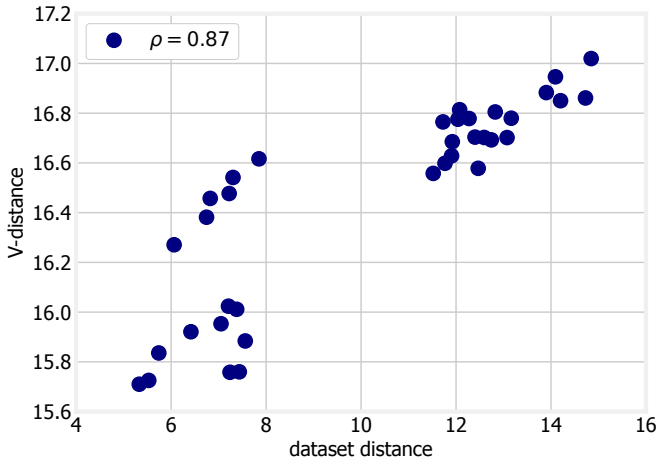

Fig. 7. Correlation between dataset distance and V-distance: each marker represents a pair of datasets. The marker's position along the x- and y-axes are equal, respectively, to the dataset distance and V-distance between the corresponding datasets.

information about local datasets and their features. Remarkably, the V-distance metric (7) results in a performance that is only marginally worse, in spite of the fact that this metric makes use of (and transfers between nodes) a much smaller amount of information. Even more importantly, the difference tends to disappear as cooperation graphs become sparser, i.e., as operational conditions for cooperation become more challenging.

In summary, Fig. 4 suggests that **it is possible (i) to remove a large fraction of the edges of the graph without significantly affecting performance, and (ii) to use efficient, privacy-preserving techniques to choose the edges to remove without incurring additional performance losses.**

In Fig. 5, we show the cooperation graphs obtained when the target number of edges is set to 18; this allows us to understand how different metrics influence the decisions made by PickEdge and the overall learning performance. We can observe a striking difference between the decisions yielded by the spectral metric (left plot, in grey) and the other two metrics (center and right plots): only considering the spectral gap always results in graphs that are as close to regular as possible. Indeed, regular graphs result in higher spectral gaps (mesh graphs have the highest possible gap). All other factors, e.g., the characteristics of the local datasets, are ignored; as mentioned earlier, this contributes to the lower performance shown in Fig. 4.

Concerning the dataset distance and V-distance metric, the resulting graphs (center and right plots in Fig. 5, denoted in red and blue, respectively) are clearly not regular. This is due to the fact that both these metrics seek to improve performance by utilizing network nodes with larger and/or better-quality local datasets, achieving the good performance highlighted in Fig. 4. Even more interestingly, the two graphs are quite similar to each other, which suggests that the two metrics lead to similar decisions.

This is confirmed by Fig. 6, highlighting that the dataset distance and V-distance metrics result in the same decision (i.e., activate or not) for the vast majority of edges. The resulting cooperation graphs overlap for more than 77% (thick, green edges in the figure). This confirms our intuition that the V-distance metric captures much of the same information as the more complete – but more onerous and less privacy-preserving – dataset distance one, and leads to decisions that not only yield similar performance, but are similar to each other.

Last, in Fig. 7 we look at the dataset distance and V-distance metrics themselves, and ascertain to which extent they are correlated. In the plot, each marker represents a pair of datasets; the marker's position along the x- and y-axes are equal, respectively, to the dataset distance and V-distance between the corresponding datasets. We can observe that the

correlation coefficient between the two metrics (denoted by ρ in the plot) is rather high, namely, 0.87. More importantly, it often happens that, given three datasets X_0, X_1, and X_2, if X_1 is closer to X_0 than X_2 according to one metric, then the same happens according to the other one. Specifically, this happens for 92% of the possible dataset triples. Considering the structure of Alg. 1, and especially Line 4, this guarantees that the decisions yielded by different metrics are, in fact, the same.

VI. Related Work

Decentralized learning is gaining popularity due to the limitations of distributed learning paradigms such as federated learning – in particular, the need of a central controller.

The studies in [12] compare decentralized learning gossip-based algorithms against federated learning algorithms in a variety of scenarios, showing that the best version of both schemes achieves similar results. In fact, [13] theoretically demonstrates that decentralized learning has the potential to surpass the performance of federated learning. This clearly underlines the potentiality of decentralized learning, whose optimization represents indeed a highly promising direction towards learning paradigms that aim at leveraging the capabilities of different nodes.

It is worth noting that decentralized learning was conceived to solve the same problem as federated learning: to perform a task in a distributed scenario to converge into a global point. With respect to how nodes interact between them, relevant works assume that node updates are sent to all neighbours [14]. Leveraging the good performance of gossip-based algorithms to efficiently disseminate information between network nodes, other studies employ them to distribute node updates [15]. Due to the lack of a central controller, decentralized learning allows for a different vision of the problem, i.e., focusing on the individual benefit of the nodes instead of looking for a common global model. This approach has been investigated in [16], which also presents an attention mechanism to adapt the problem to new conditions, e.g., the presence of additional learning nodes.

Another important aspect in decentralized learning is the use of non-i.i.d. data, which may be difficult to cope with, as in the case of any distributed learning paradigm [17]. Existing works address this issue from different perspectives, e.g., by assigning weights to the received information [16], modifying gradients as the training process approximates low loss values [18], or varying the communication frequency between learning nodes depending on specific data indicators [17].

Optimizing energy, or, more in general, cost and system resources is also critical. From the viewpoint of the network, reducing the number of data transmissions can save a substantial amount of bandwidth [17]. However, it is crucial to analyze the existing trade off between system efficiency and ML model consensus [19].

Recently, a novel trend has emerged, which is has been explored under different learning paradigms. The idea is to control the interaction between learning nodes by acting on the physical or logical network topology. For example, [4] presents an analysis of cooperation among candidate learning nodes in several distributed learning scenarios and provides an algorithm that optimizes time and system resources. Additionally, [20] aims to redefine the topology of Urban Air Mobility (UAM) vehicles to assist the performance of federated learning.

Some works have envisioned integrating this concept within decentralized learning. Specifically, it has been demonstrated analytically that the logical topology may have a significant impact on the performance of decentralized learning [21]. In fact, [22] proves that a ring communication topology allows increasing the number of nodes without damaging the spectral gap, which affect the convergence of a model training. Additionally, [23] presents a hierarchical framework that selects learning nodes by applying a clustering method that accounts for the position of the network nodes.

Our approach contributed to the optimization of the learning nodes' logical topology. Specifically, we investigate how reducing the number of cooperating learning nodes can substantially increase the efficiency of decentralized learning while maintaining the process efficacy.

VII. Conclusions and Future Work

We have targeted the problem of decentralized learning in mobile networks, where the benefits of cooperation between learning nodes must be balanced against the resulting overhead and cost. In this context, we have proposed an efficient and effective algorithm, called PickEdge, able to generate near-optimal cooperation graphs using any metric expressing how desirable the cooperation between two nodes is.

We have proposed, discussed, and evaluated three different metrics, each exhibiting a different trade-off between the information that is needed and the resulting performance. Through our performance evaluation, we have found that privacy-preserving metrics leveraging singular value decomposition to estimate the distance between local datasets yield essentially the same performance as less-efficient, non-privacy-preserving metrics that directly measure the distance itself.

Future work will focus on expanding the scope of our performance evaluation to additional deep neural networks architectures and scenarios, as well as comparing PickEdge's performance against benchmarks that exploit reinforcement learning approaches.

Acknowledgment

This work was supported by the SNS-JU-2022 Grant No. 101095890 (PREDICT-6G project) and Grant No. 101096379 (CENTRIC project) under the EU's Horizon Europe research and innovation programme.

References

[1] J. Konečnỳ, B. McMahan, and D. Ramage, "Federated optimization: Distributed optimization beyond the datacenter," *arXiv preprint arXiv:1511.03575*, 2015.

[2] J. Ren, G. Yu, and G. Ding, "Accelerating dnn training in wireless federated edge learning systems," *IEEE Journal on Selected Areas in Communications*, 2020.

[3] F. Malandrino, G. Di Giacomo, A. Karamzade, M. Levorato, and C. F. Chiasserini, "Matching DNN compression and cooperative training with resources and data availability," in *IEEE INFOCOM*, 2023.

[4] F. Malandrino, C. F. Chiasserini, N. Molner, and A. De La Oliva, "Network support for high-performance distributed machine learning," *IEEE/ACM Transactions on Networking*, 2022.

[5] G. Neglia, G. Calbi, D. Towsley, and G. Vardoyan, "The role of network topology for distributed machine learning," in *IEEE INFOCOM*, 2019.

[6] D. Anguita, A. Ghio, L. Oneto, X. Parra, J. L. Reyes-Ortiz *et al.*, "A public domain dataset for human activity recognition using smartphones," in *ESANN*, 2013.

[7] A. N. D. Model, "Learning framework for edge ai," *IEEE Internet of Things Journal*, 2020.

[8] Z. Allen-Zhu, Y. Li, and Z. Song, "A convergence theory for deep learning via over-parameterization," in *ICML*, 2019.

[9] R. K. Iyer and J. A. Bilmes, "Submodular optimization with submodular cover and submodular knapsack constraints," in *NeurIPS*, 2013.

[10] V. Vu, "Random discrete matrices," in *Horizons of combinatorics*. Springer.

[11] K. Tikhomirov and P. Youssef, "The spectral gap of dense random regular graphs," *The Annals of Probability*, 2019.

[12] I. Hegedűs, G. Danner, and M. Jelasity, "Decentralized learning works: An empirical comparison of gossip learning and federated learning," *Journal of Parallel and Distributed Computing*, 2021.

[13] X. Lian, C. Zhang, H. Zhang, C.-J. Hsieh, W. Zhang, and J. Liu, in *NeurIPS*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds.

[14] A. Lalitha, S. Shekhar, T. Javidi, and F. Koushanfar, "Fully decentralized federated learning," in *NeurIPS BDL workshop*, 2018.

[15] Y. Lu and C. D. Sa, "Decentralized learning: Theoretical optimality and practical improvements," *Journal of Machine Learning Research*, 2023.

[16] S. Li, T. Zhou, X. Tian, and D. Tao, "Learning to collaborate in decentralized learning of personalized models," in *IEEE/CVF CVPR*, 2022.

[17] K. Hsieh, A. Phanishayee, O. Mutlu, and P. Gibbons, "The non-IID data quagmire of decentralized machine learning," in *ICML*, 2020.

[18] Y. Shi, L. Shen, K. Wei, Y. Sun, B. Yuan, X. Wang, and D. Tao, "Improving the model consistency of decentralized federated learning," in *ICML*, 2023.

[19] W. Liu, L. Chen, and W. Zhang, "Decentralized federated learning: Balancing communication and computing costs," *IEEE Transactions on Signal and Information Processing over Networks*, 2022.

[20] K. Xiong, R. Wang, S. Leng, W. Che, C. Huang, and C. Yuen, "Ris-empowered topology control for distributed learning in urban air mobility," *arXiv preprint arXiv:2403.05133*, 2024.

[21] T. Vogels, H. Hendrikx, and M. Jaggi, "Beyond spectral gap: the role of the topology in decentralized learning," in *NeurIPS*.

[22] W. Zhang, X. Cui, A. Kayi, M. Liu, U. Finkler, B. Kingsbury, G. Saon, Y. Mroueh, A. Buyuktosunoglu, P. Das, D. Kung, and M. Picheny, "Improving efficiency in large-scale decentralized distributed training," in *IEEE ICASSP*, 2020.

[23] L. Yang, Y. Lu, J. Cao, J. Huang, and M. Zhang, "E-tree learning: A novel decentralized model learning framework for edge ai," *IEEE Internet of Things Journal*, 2021.