A Key Performance Indicator to Analyze Swarm Learning Performances with EHR

A key performance indicator to analyze swarm learning performances with EHR

Riccardo Scheda*
Department of Computer Science
University of Verona - Italy
riccardo.scheda@univr.it

Matteo Mantovani
Department of Computer Science
University of Verona - Italy
matteo.mantovani@univr.it

Carlo Combi
Department of Computer Science
University of Verona - Italy
carlo.combi@univr.it

Stefano Diciotti
Department of Electrical,
Electronic, and Information Engineering
University of Bologna - Italy
stefano.diciotti@unibo.it

***Abstract*—Swarm Learning (SL) has been recently proposed for distributed learning, where a group of individual centers perform a synchronized training. Unlike traditional machine learning models that rely on a central server, swarm learning distributes the learning process across multiple nodes. Each node independently processes data and contributes to the overall learning task. This collaboration allows the swarm to benefit from individual nodes' different data. Unlike federated learning, here model parameters are not handled by a central server but are randomly handled across each individual node. The intrinsic attention of swarm learning to data privacy makes it suitable for distributed healthcare analysis, where a clinical center wants to benefit from all the other ones in the swarm network. However, the benefit for a single center or for the whole network could vary depending on data distribution.**

In this paper, we want to analyze the performance of the swarm learning in a network with multiple nodes, where different data distribution scenarios are taken into account. This analysis will show the gain of the whole swarm network and a specific (reference) node, focusing on scenarios where this node has a different amount of data with respect to the other nodes. To perform a more analytical analysis, we introduce a new Key Performance Indicator (KPI) to measure such gain. We then applied this method using ICU data extracted from the MIMIC EHR database and discussed the results obtained by analyzing 5 nodes with different data distribution scenarios.

***Index Terms*—swarm learning, performance analysis, KPI, EHR, distributed computing**

I. Introduction

In the last decades, the clinical domain has witnessed an increasing integration of digital technologies to improve patient care and, more generally, how healthcare is managed and experienced. Nowadays, many hospitals and clinical organizations adopt digital solutions, and many clinical datasets are available to researchers and healthcare professionals. The analysis of those data is fundamental for decision-making processes related, for example, to resource allocation, patient care, disease progression prediction, and treatment protocols. However, there could be situations where the amount of data collected by a single center is not sufficient (e.g., for rare disease analysis), and it could be beneficial to perform an analysis across multiple servers. Nevertheless, many clinical centers cannot/will not share data with other centers (e.g., for privacy reasons). To address this issue, there were developed privacy-preserving distributed algorithms, such as Secure Multiparty Computation (MPC) [1], homomorphic encryption [2], and, more recently, the federated learning [3]. With federated learning, the training of a model is performed across multiple nodes while ensuring that the data of each node remains decentralized (i.e., each node does not have to share its data with the other nodes). This is obtained by training local models on their own data and exchanging parameters (e.g., the weights in a deep neural network) periodically between all the nodes to obtain a global model. However, with federated learning, a central server is identified to handle the sharing and synchronization of the parameters, and this aspect could also decrease the the fault tolerance [4].

Swarm Learning (SL) is a recently proposed technique that addresses this issue, and it allows to perform a distributed and synchronized training to solve a problem without the need for a permanent centralized control [4]. Swarm Learning relies on a specific concept of node, which is a component of the framework that performs a specific function and collaborates with other nodes in the network, regardless of their physical location on one or multiple servers. Each node independently processes data and contributes to the overall learning task. This collaboration allows the swarm to benefit from individual nodes' different data sample information. The true potential of swarm learning relies on addressing privacy concerns by allowing data to remain on individual nodes. Also, the sharing and synchronization of the parameters relies on an epoch leader, which is a node randomly chosen at each synchronization step. In the published paper [4] presenting swarm learning, authors investigated different configurations of the training processes, with some variation in the number

*previously affiliated with *Department of Electrical, Electronic, and Information Engineering*, *University of Bologna* - Italy

of samples between the nodes, and some limited study of distribution of data over heterogeneous diseases. Moreover, they do not compare SL-trained models with the central server model, where the model is trained on the whole dataset in a single server. Finally, the authors used only a simple average as a merging algorithm without analyzing different configurations with different weights for the nodes. Other recent works focused on the swarm learning performance and application in the healthcare domain [5], [6], but there are no studies that systematically analyzed the swarm learning performance when the amount of data is different between the nodes. For example, let us consider a swarm network composed of 5 different hospitals, where 4 of them have a very similar amount of data. However, the amount of data on the fifth node could differ from the others within the swarm network. Does a different amount of data from this center benefit the swarm network, and how? What are the advantages or disadvantages for this node to be part of the swarm network, compared to locally training a model using only its data?

In this paper, we want to answer these questions and focus on the swarm learning performance by systematically exploring various data configurations between the nodes. This investigation analyzes how a swarm network adapts and performs across diverse scenarios in which a node owns different data distributions with respect to the other nodes. We propose a new Key Performance Indicator (KPI) to analitically analyze the gain of a reference node and the swarm network based on different data distributions. Finally, we test our proposed approach using the ICU data from the MIMIC-III dataset and discuss our results.

II. Background and Related Works

Swarm Learning (SL) was first introduced in 2021 by Warnat-Herresthal et al. [4] as an evolution of federated learning that does not need a central coordinator and it relies on a blockchain-based peer-to-peer networking and coordination. That paper focused on the feasibility of using SL to develop disease classifiers using distributed data, choosing four use cases of heterogeneous diseases of COVID-19, tuberculosis, leukemia, and lung pathologies, with different sample sizes, ranging from 899 samples of a whole-blood-derived transcriptome dataset, to 47,000 X-ray images dataset. They analyzed some limited scenarios of uniform and non-uniform distributions for each of the datasets between the nodes. In the paper, the authors showed that in their configuration the SL model almost always outperforms the performance of the models trained locally in each node. However, they do not always compare SL-trained models with the central server model, where the model is trained on the whole dataset in a single server. After this initial work, other groups started working on SL, and in particular in the clinical domain. In [5], authors trained deep learning models with swarm learning using multicentric datasets of gigapixel histopathology images from over 5,000 patients. Here, the authors compared swarm learning models with models trained on the whole dataset in a single server. In [6], authors collected digital whole-slide images (WSIs) of tissue section samples. They included four cohorts of patients with gastric cancer from four countries, obtaining more than 2000 samples. Three of these cohorts were used as training cohorts, and one was used as the testing cohort. It is interesting to note that, in both of the works by Saldanha et al. and Warnat-Herresthal et al., the central model metrics consistently outperform or, in some cases, perform on par with SL models. The focus of all of these papers was on different topics than a systematic performance analysis of the SL network under different data distribution scenarios, which still lacks as of today.

A. Artificial Neural Network

Artificial Neural Networks (ANNs) are a large branch of computational systems included in the field of Artificial Intelligence. ANNs include several types of networks, each one built for a specific problem or a particular type of data. For example, Convolutional Neural Networks (CNNs) are networks that have built-in convolution functions, which are well-suited for tasks such as object detection, image classification, or similar tasks related to image-based data [7]. Inspired by biological brain structures, neural networks are composed of layers of nodes (i.e., neurons) and links. Connecting several layers of neurons, each node of the network receives an input, processes it with a predetermined function, and produces an output, which is sent to the next layer of nodes [8]. The last layer of nodes produces the desired output of a task, which can be a binary or multi-class classification or a regression task. The learning process of the network comes through an algorithm called backpropagation [7], which progressively adjusts the weights of the nodes through the layers, given the errors made during the prediction. In this work, we use a fully connected neural network, which is a type of neural network well-suited for tabular data.

B. MIMIC-III

Medical Information Mart for Intensive Care (MIMIC)-III [9] is a publicly-available relational database, and it contains data related to Intensive Care Unit (ICU) patients who stayed in critical care units of the Beth Israel Deaconess Medical Center between 2001 and 2012. These data are deidentified and associated with more than $60\,000$ admissions for $46\,000$ unique patients. The database includes different kinds of information to compose a complete medical record of a patient. MIMIC-III includes vital sign measurements (automatically taken or manually reported), demographics, laboratory test results, prescriptions, procedures, diagnosis, caregiver notes, and many other valuable information to represent a complete description of a patient's stay accurately.

C. HPE Swarm Learning software

Hewlett-Packard Enterprise (HPE) developed a freely available swarm learning software, named HPE Swarm Learning Community Edition*, that allows to train collaborative models between different servers without sharing the training data.

*https://github.com/HewlettPackard/swarm-learning

This is achieved by individual nodes sharing model weights derived from training the model on the local data of each node. The SL process begins with enrollment on the swarm smart contract by each node. Next, nodes proceed to train a local copy of the model iteratively over multiple epochs, for a fixed number of iterations. After reaching the number of iterations, each node exports its model parameters and sends them to the epoch leader, which is chosen randomly at each synchronization step between all the nodes. Then, the epoch leader merges all the models' parameters from each node and sends merged parameters to the other peers. Then, each node proceeds to train the model, until the fixed iteration number is reached again. At the end of all the synchronization steps, a final swarm model is obtained. Each of the nodes receives the swarm merged model.

Weights' merging process is done through a merging algorithm: this algorithm is not customizable, but the HPE official package proposes three different options: weighted average, coordinate median, and geometric median.

The maximum amount of nodes available in the HPE Swarm Learning Community edition is 5. However, there is complete flexibility in the logical position of these nodes. In fact, it is possible to have a single node for each connected server, as well as multiple nodes running on a single server, or a combination of both scenarios.

III. The proposed key performance indicator for swarm learning

The intrinsic nature of the swarm learning framework, with its distributed learning and attention to data privacy, makes it a natural candidate for performing distributed analysis in healthcare across multiple centers. In this section, we want to analyze systematically the effects of a single - *reference* - node when its data amount differs from the others in the swarm network. In particular, we wanted to focus on the impact on both the swarm network and the node itself in terms of increased/decreased performance with different data distribution scenarios between the reference node and the others. This analysis simulates different real-world scenarios in which the amount of data for each center is different. To do so, we considered the reference node with a given percentage of the training set, uniformly distributing the remaining set in the other nodes. The evaluation is repeated many times varying the percentage of the training set for the reference node and uniformly distributing the remaining percentage of the training set in the other nodes. In particular, we compare the performances of the swarm model (p_s) and of the locally trained models (p_i) for each node i. All the models' performances are evaluated over the same *test* data for all the nodes.

To evaluate quantitatively the impact of a different data distribution between one reference node and other nodes, we introduce a Key Performance Indicator (KPI), $\kappa \in [-1, 1]$, defined in the equation:

$$\kappa = \sum_{i=1}^{N} w_i (p_s - p_i) \tag{1}$$

where:

- N is the total number of nodes within the swarm network ($N \geq 2$);
- w_i is the weight of the i-th node, $w_i \in [0, 1]$, which corresponds to the percentage of the training set within that node;
- p_s represents the performance of the swarm model $p_s \in [0, 1]$;
- $p_i \in [0, 1]$ describes the model performance of the i-th node, trained on its local data.

Positive values of κ correspond to an overall positive contribution of all the nodes in the swarm network with the considered weights.

Equation (1) describes the overall gain obtained by the network when using swarm learning instead of locally trained models. Nevertheless, to emphasize the contribution of the reference node, we may distinguish its contribution from the contribution of the other nodes. So, we modify the previous equation defining the gain G_r of the performance of the reference node, namely p_r:

$$G_r = p_s - p_r \tag{2}$$

The gain G_o of the performance of the other p_i nodes could be expressed as:

$$G_o = \sum_{i=1, i \neq r}^{N} \frac{p_s - p_i}{N - 1} \tag{3}$$

Because we want to analyze the *average* performance of the other nodes, namely p_o, the previous expression could be transformed as follows:

$$G_o = p_s - \sum_{i=1, i \neq r}^{N} \frac{p_i}{N - 1} = p_s - p_o \tag{4}$$

To systematically measure the performance of the swarm network, where we change the data distribution, we must add some constraints to minimize the influence on κ. First of all, we do not change the distribution of the classes between the different nodes. Also, in order to correctly observe variations in performances given by the reference node, we equally split and distributed the remaining data within the other nodes.

For this reason, only the weight of the reference node w_r differs, while the weights of all the other nodes w_{o_i} are kept equal:

$$w_{o_1} = w_{o_2} = \cdots = w_{o_{N-1}} = w_o \tag{5}$$

And the sum of all the weights is:

$$w_r + (N - 1) w_o = 1 \tag{6}$$

So, equation (1) becomes:

$$\kappa = w_r (p_s - p_r) + (N - 1) w_o (p_s - p_o) \tag{7}$$

This expression could also be written as:

$$\kappa = w_r G_r + (N - 1) w_o G_o \tag{8}$$

IV. Experimental evaluation

In this section, we show the experimental evaluation of the principles described in the previous section; that is, we investigate swarm learning with different configurations to analyze variations in performance.

In order to minimize the variation on κ due to factors external to data distribution, we added some constraints. As mentioned before, we do not change the distribution of the classes between the different nodes. Moreover, starting from a specific extraction from the MIMIC-III database (as described below), we randomly select 70% of the dataset as *train* set. The remaining 30% of the dataset is used as *test* set, in order to compare the metrics results for all the nodes in the swarm network with the same data. The reference node data distribution varies from 2.5% to 90% of the original training set, and we equally split and distributed the remaining data within the other nodes. That is, the remaining training data was uniformly split into non-overlapping subsets between the remaining nodes, obtained by performing stratified splitting. Table I accurately represents all the different configurations and weights used in our experimental evaluation. Each node trained the model and periodically merged the parameters with the other nodes. When the swarm training is finished, each node tested its model on the test set. We ran 10 different trainings using random data shuffles with different random seeds for each configuration, and we averaged the resulting performance.

Finally, let us note that the HPE Swarm Learning software represents the weight only as integer values, in the range $[1 - 100]$ (see the official documentation[*]). Due to this limitation, we have to transform the equation 8 of Section III as follows:

$$\kappa = \frac{w_r G_r + (N-1) w_o G_o}{w_r + (N-1) w_o}. \tag{9}$$

Fold size (%)					Weights				
Reference Node	Other nodes				Reference Node	Other nodes			
2.5	24.375	24.375	24.375	24.375	2	24	24	24	24
5	23.75	23.75	23.75	23.75	5	23	23	23	23
10	22.5	22.5	22.5	22.5	10	22	22	22	22
20	20	20	20	20	20	20	20	20	20
30	17.5	17.5	17.5	17.5	30	17	17	17	17
40	15	15	15	15	40	15	15	15	15
50	12.5	12.5	12.5	12.5	50	12	12	12	12
60	10	10	10	10	60	10	10	10	10
70	7.5	7.5	7.5	7.5	70	7	7	7	7
80	5	5	5	5	80	5	5	5	5
90	2.5	2.5	2.5	2.5	90	2	2	2	2

TABLE I: Percentages of the dataset and weights of the nodes in different configurations. Weights have integer values since the HPE Swarm Learning software allows only integer values for the weighted mean.

[*]https://github.com/HewlettPackard/swarm-learning/blob/master/docs/User/How_to_Swarm_enable_an_ML_algorithm.md

A. System configuration and software

For our analysis, we adopted the HPE Swarm Learning Community software described in Section II. As discussed in the previous sections, we want to analyze the performance behavior of a swarm network with different data distributions between a reference node and all the others. Thus, we adopted the *weighted average* merging option to reflect the percentage of distribution of the dataset across the nodes. Consequently, we gave each node a weight value that corresponds to the percentage of the training set within that node.

In this work, we analyzed different data distributions across 5 nodes, which is the maximum number of nodes allowed by the community edition of the HPE software. We decided to use the maximum number of nodes available to simulate a real-world scenario where 5 different centers organize and join a swarm network. It is important to note that the performance analysis discussed in this paper focused on the nodes. Thus, the results obtained are independent of the physical location of the nodes. For this reason, and to speed up the synchronization process, we decided to run all the nodes within a single server. It is a Linux workstation equipped with a 16-core AMD Opteron(tm) Processor 4386 CPU and 16 GB RAM. It runs on Ubuntu 22.04.2 LTS and with HPE Swarm Learning Community edition 2.2.0, with a computation time of about 26.5 hours for the entire experimental evaluation with all the different data distributions.

Since the HPE Swarm Learning package is actually compatible only with deep learning models, we chose to build an Artificial Neural Network (ANN). The neural network consists of one input layer, three hidden layers, and one output layer. The three hidden layers are densely connected and consist of 16 nodes each, with a rectified linear unit activation function (ReLU [10]). The output layer is densely connected and consists of one node and a sigmoid activation function. The model is configured for training with Adam optimization [11] and to compute the binary cross-entropy loss between true and predicted labels. Training is performed on the same model both for the individual nodes on their local data and in the SL framework. The model is trained over 25 epochs, with a batch size of 64. The task is a binary classification, so we chose the area under the receiver operating characteristic curve (AUROC) to measure the model performance.

B. The dataset

To test and validate our method, we analyze clinical data derived from the MIMIC III dataset [9] with the goal of identifying patterns predictive of sepsis diagnosis. Because of that, it is not necessary to use all the data contained in MIMIC, and we have to perform an ETL (Extract, Transform, Load) process to retrieve only the portion of data we need and in the proper format. For the sepsis prediction, we extracted the following features for each admitted patient:

- Vital signs. From the CHARTEVENTS table, we retrieved heart rate, diastolic and systolic blood pressure, white blood cells, and body temperature across the admission.

For each vital sign, we extracted the minimum and maximum values recorded during a patient's hospitalization, as two independent features. The numerical values were discretized to low, normal, and medium ranges, using the thresholds shown in Table II.

- Procedures (true\false) from PROCEDURES_ICD.
- Medications, from the INPUTEVENTS_MV table, that are: *vancomycin, piperacillin/tazobactam, ciprofloxacin, epinephrine, norepinephrine, vasopressin, dopamine*, metoprolol, potassium chloride, phenylephrine, omeprazole (prilosec), and pantoprazole (protonix). Here, we selected some medications recommended for sepsis treatment according to international guidelines [12], marked in *italic*, and other more generic medications. This is to verify the goodness of the prediction.
- Sepsis (true\false), retrieved from the DIAGNOSES_ICD table and using the associated ICD codes.

At the end of the E.T.L. process we obtained data for 21 875 patients, 2804 of them with sepsis. For the experimental evaluation, we had 15 307 samples in the training set and 6568 samples for the test set, keeping the same proportions of patients with sepsis.

	Heart Rate	**DBP**	**SBP**	**W.B.Cs**	**Body Temp.**
Low	< 60	< 60	< 90	< 4.0	< 36.0
High	> 90	> 90	> 140	> 12.0	> 38.0

TABLE II: Thresholds used to discretize the considered vital signs in *low* or *high*. The other values are labeled as *medium*.

C. Results

First of all, we calculated the average AUROC for the central model. It is obtained with the model trained on a single server with the whole dataset, over ten repetitions with random shuffles of the training data with different random seeds, and its AUC is 0.854 ± 0.004 (average $\pm$ standard deviation). The central model, as well as the swarm and local models, were tested on the same test set with unseen data.

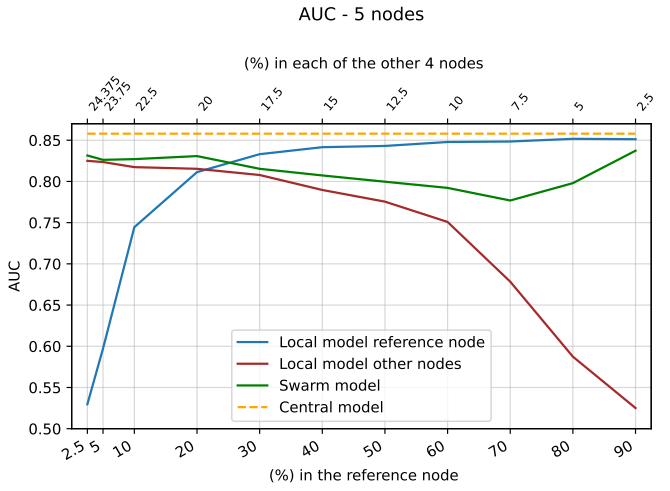

Fig. 1: Average AUC performance of the models with different data distribution scenarios. In particular, the figure shows the models for the reference node, the other nodes, the swarm network, and the central one.

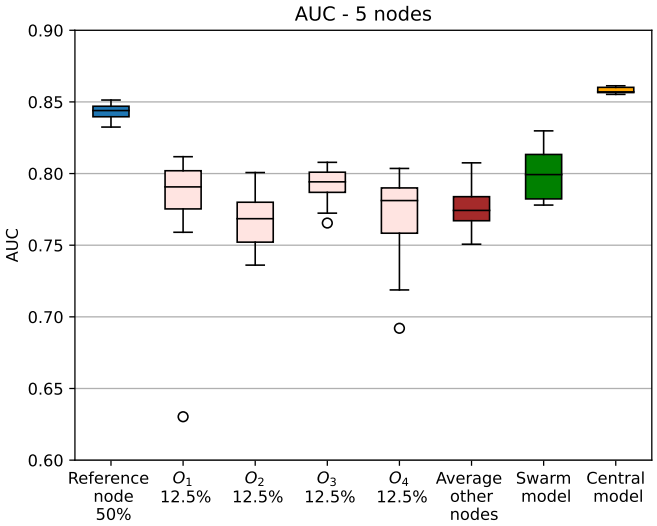

Fig. 2: Box plots of AUROC for the configuration (50%, 12.5%, 12.5%, 12.5%, 12.5%). The black line within the boxplots represents the median; box limits, 1st and 3rd quartiles; remaining dots: outliers.

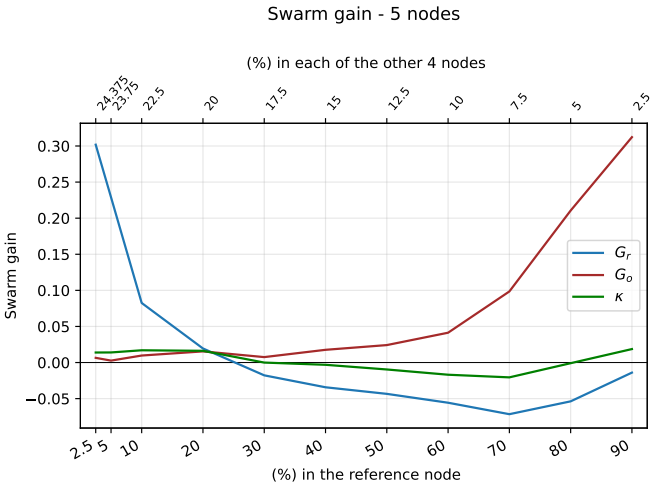

Fig. 3: Plot of KPI κ varying the distribution of the dataset among the nodes. Blue line: gain of the reference node G_r; brown line: gain of the remaining nodes G_o; green line: KPI κ, which represents the overall gain in performance for the nodes for being part of the swarm network.

Then, as shown in Figure 1, we calculated the AUC for the models of the reference node, of the other nodes, and the swarm model, varying the size of the distribution for the reference node from 2.5% to 90% of the entire training set. We can observe that the performance of the reference node increased as the percentage of its data distribution increased, almost reaching the performance of the central model.

Complementary, the average performance of the other nodes models decreased as the portion of the dataset decreased. Intriguingly, the performance of the swarm model reached its peak performance when all the 5 nodes had the same data distribution (i.e., 20%), even though they were trained on different instances of data. The lowest value of AUC for the swarm model was obtained when the reference node had the 70% of the data, and the remaining 30% was distributed as 7.5% for the remaining four nodes.

In Figure 2, we illustrated an example of the data distribution used in our analysis, with 50% of the data for the reference node and 12.5% for each other node. We can see that the performance of the central model outperformed the model performances of all the nodes, trained on their local data. The reference node outperformed the model performance of the other nodes, while the swarm model performance was only slightly better. The figure also remarks that each *other* node could have a different performance, but here we considered the average performance of all of them.

Figure 3 shows the values of κ under different data distribution conditions, and it also shows the gain for the reference node (G_r) and the other nodes (G_o). In this figure, we can observe many interesting behaviors. First of all, κ had positive values in the range $[2.5\% - 30\%]$ and above 80%, but the values across all the data distributions were very close to zero. This could indicate that the swarm network is fairly stable, and even a node with the 90% of the data does not unbalance the performance of the network (positively or negatively). On the other hand, the reference node had a positive gain until its distribution was lower or equal than that of the other nodes. However, it is interesting to note that all the other nodes always benefit from being part of the swarm network, regardless of the amount of data on the reference node, as G_o is always greater than zero.

V. Conclusions

In this work, we analyzed different configurations of the dataset in a swarm network with five nodes. In particular, we focused on scenarios in which a reference node has different portions of the training set, starting from 2.5% to 90% of the original training set. We defined a KPI, namely κ, to quantitatively evaluate the impact of different data distributions on the reference node w.r.t. the swarm network, while keeping the remaining data equally split and distributed across all the other nodes. The study of κ also allowed us to observe the gain in performance for all the nodes in the network when using swarm learning instead of locally trained models. When the data distribution across the nodes was equal (e.g., 20% of the training set in each node), the AUC of the swarm model outperformed the individual AUC of each node model trained with local data. When the percentage of the reference node increased, we could observe that the performance of the other nodes decreased consequently. The swarm model AUC behavior reflected the weighted performance of all the nodes involved in the network. It is interesting to note that, when the majority of the original training set (e.g., $80/90\%$) was located in the reference node, the swarm model performance verged to the performance of the reference node. On the opposite, when the majority of data was located in the other nodes, the swarm model performance tended to their performance.

In the future, we will explore different scenarios while introducing other constraints. For example, it could be interesting to analyze the differences in performance while changing the number of nodes. This new evaluation may also help us to study the impact of a new node joining the swarm network. Also, we plan to study the behavior of the network when the reference node has different distributions for the classes w.r.t. the other nodes.

Acknowledgment

The research leading to these results has received funding from the European Union—NextGenerationEU through the Italian Ministry of University and Research under PNRR—M4C2-I1.3 Project PE_00000019 "HEAL ITALIA" to the authors CUP B33C22001030006. The views and opinions expressed are those of the authors only and do not necessarily reflect those of the European Union or the European Commission. Neither the European Union nor the European Commission can be held responsible for them.

References

[1] R. Canetti, U. Feige, O. Goldreich, and M. Naor, "Adaptively secure multi-party computation," in *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, 1996, pp. 639–648.

[2] A. C. Yao, "Protocols for secure computations," in *23rd annual symposium on foundations of computer science (sfcs 1982)*, 1982, pp. 160–164.

[3] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," 2017.

[4] S. Warnat-Herresthal, H. Schultze, K. L. Shastry, S. Manamohan, S. Mukherjee, V. Garg, R. Sarveswara, K. Händler, P. Pickkers, N. A. Aziz *et al.*, "Swarm learning for decentralized and confidential clinical machine learning," *Nature*, vol. 594, pp. 265–270, 2021.

[5] O. L. Saldanha, P. Quirke, N. P. West, J. A. James, M. B. Loughrey, H. I. Grabsch, M. Salto-Tellez, E. Alwers, D. Cifci, N. Ghaffari Laleh *et al.*, "Swarm learning for decentralized artificial intelligence in cancer histopathology," *Nature Medicine*, pp. 1232–1239, 2022.

[6] O. L. Saldanha, H. S. Muti, H. I. Grabsch, R. Langer, B. Dislich, M. Kohlruss, G. Keller, M. van Treeck, K. J. Hewitt, F. R. Kolbinger *et al.*, "Direct prediction of genetic aberrations from pathology images in gastric cancer with swarm learning," *Gastric cancer*, 2023.

[7] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[8] I. J. Goodfellow, Y. Bengio, and A. C. Courville, *Deep Learning*, ser. Adaptive computation and machine learning. MIT Press, 2016.

[9] A. E. Johnson, T. J. Pollard, L. Shen, L.-w. H. Lehman, M. Feng, M. Ghassemi, B. Moody, P. Szolovits, L. Anthony Celi, and R. G. Mark, "Mimic-iii, a freely accessible critical care database," *Scientific data*, vol. 3, pp. 1–9, 2016.

[10] A. F. Agarap, "Deep learning using rectified linear units (relu)," *CoRR*, vol. abs/1803.08375, 2018.

[11] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015.

[12] L. Evans, A. Rhodes, W. Alhazzani, M. Antonelli, C. M. Coopersmith, C. French, F. R. Machado, L. Mcintyre, M. Ostermann, H. C. Prescott *et al.*, "Surviving sepsis campaign: international guidelines for management of sepsis and septic shock 2021," *Critical care medicine*, vol. 49, no. 11, pp. e1063–e1143, 2021.