Title: *On Assessing Heterogeneity Management Solutions in Federated Learning Systems*

Authors: *Luciano Baresi, Tommaso Dolci, and Iyad Wehbe*

Published in: *2024 IEEE/ACM 17th International Conference on Utility and Cloud Computing (UCC)*

Abstract: *Federated machine learning, often referred to as Federated Learning (FL), is a convenient way to distribute machine learning tasks at different nodes. FL helps keep privacy and may save bandwidth, but comes with some peculiarities. One key problem is the management of heterogeneity to obtain robust and convergent models. Among the many dimensions, this paper tackles heterogeneity by considering node selection and workload optimization and assesses some alternative solutions to better understand the pros and cons. We implemented them in Flower, a flexible user-friendly FL framework, and evaluated their performance compared to baseline techniques on an MLP model and using the MNIST dataset with different degrees of heterogeneity in data distribution. The results obtained provide interesting insights on their effectiveness, convergence speed, and stability. The goal is also to encourage the community to extend the experiments and play with different strategies, features, and characteristics.*

DOI: *10.1109/UCC63386.2024.00080*

On Assessing Heterogeneity Management Solutions in Federated Learning Systems

Luciano Baresi, Tommaso Dolci, and Iyad Wehbe
Politecnico di Milano, Dipartimento di Elettronica, Informazione e Bioingegneria
via Golgi, 42 - 20133 Milano (Italy) {name.surname}@polimi.it

***Abstract*—Federated machine learning, often referred to as Federated Learning (FL), is a convenient way to distribute machine learning tasks at different nodes. FL helps keep privacy and may save bandwidth, but comes with some peculiarities. One key problem is the management of heterogeneity to obtain robust and convergent models. Among the many dimensions, this paper tackles heterogeneity by considering node selection and workload optimization and assesses some alternative solutions to better understand the pros and cons. We implemented them in Flower, a flexible user-friendly FL framework, and evaluated their performance compared to baseline techniques on an MLP model and using the MNIST dataset with different degrees of heterogeneity in data distribution. The results obtained provide interesting insights on their effectiveness, convergence speed, and stability. The goal is also to encourage the community to extend the experiments and play with different strategies, features, and characteristics.**

***Index Terms*—federated learning, heterogeneity, node selection, workload selection, assessment.**

I. Introduction

Federated learning (FL) is emerging as a paradigm for addressing privacy concerns and maybe helping save bandwidth, by supporting machine learning training and inference on distributed nodes [1]. FL offers distinct advantages over traditional ML, including reducing latency (during inference) and improving privacy (during training) by minimizing data sharing and keeping potentially sensitive information locally [2], [3]. The participating nodes retrieve the initial —-perhaps empty— model from a central server, conduct model training with their local data, and subsequently communicate the weights of the resultant model back to the server. The weights are then aggregated and the process is repeated iteratively to meet set qualities of service.

Needless to say, FL introduces its own problems, and the management of heterogeneity is a key concern [2], [4]. **Statistical heterogeneity** can impede model convergence and arises from the non-IID (non-independent and identically distributed) nature of data in different nodes, coupled with significant variations in the sizes of local data samples and influenced by individual behaviors [5], [6]. On the other hand, **system heterogeneity** encompasses variations in node attributes, such as computational capabilities, memory, energy availability, and environmental factors (e.g., network speed and reliability) [4], [7], [8]. In contrast to the relatively uniform and robust machines used in traditional approaches, FL contends with diverse and less predictable system conditions.

To address these challenges, we must consider factors such as the quality, size, and distribution of local data, computational resources, and node availability to improve node selection and resource management in the federation, mitigate the impact of slower nodes, and improve the learning process. Solutions that do not consider the evolution of the learning process ([2]), or that do not consider the heterogeneity of nodes [9], do not always pay off.

This paper studies statistical heterogeneity by considering node selection and workload optimization techniques. It evaluates the performance of four different well-known algorithms for node selection (*Dynamic Sampling*, *pow-d*, *cpow-d*, and *rpow-d*), along with *FedAvg* as baseline for comparison. It also considers system heterogeneity by assessing the performance of four workload optimization techniques (*Static Optimizer*, *Uniform Optimizer*, *Round Time Optimizer*, and *Equal Computation Time Optimizer*). We implemented all these techniques in Flower [10], a flexible FL framework that supports experimentation with a variety of learning algorithms and federation settings[1]. We then performed a thorough evaluation on an MLP model and using the MNIST dataset with different degrees of heterogeneity in data distribution. Flower offers a built-in simulation engine to facilitate experimentation and validation of diverse scenarios.

The remainder of the paper is organized as follows. Section II illustrates the strategies considered in the paper for node selection and workload optimization. Section III discusses the assessment of the different strategies. Section IV surveys the related work, and Section V concludes the paper.

II. Node selection and Workload Optimization

This section provides a brief introduction to the algorithms implemented and analyzed in this paper. These solutions are mainly well-known algorithms taken from the literature and, partly, come from our previous work.

A. Node Selection

Besides random selection, which remains a widely used solution, more appropriate selection strategies can significantly improve performance. These solutions exploit non-sensitive nodes' metadata: for example, the quality and size of local samples, availability of computational resources, or even battery life. With this information, the server can make

[1]The code and data used in this paper are available at: https://zenodo.org/records/13987517

more informed choices about node selection and speed up the convergence of training, or maintain similar performances using fewer nodes at each round, and save resources.

We consider four strategies for dynamic node selection: *Dynamic Sampling*, *pow-d*, *cpow-d*, and *rpow-d*. Flower provides a built-in implementation of *FedAvg*, and we used it as a baseline for comparison. We implemented the others as custom *Strategies* to facilitate comparison and reproducibility.

All strategies assume a framework of N nodes and a central server. Training evolves through multiple consecutive rounds $t = 1 \dots R$. Each node k has a unique local dataset D_k of size s_k. The sum of the sizes of all local datasets is $s = \sum_{k=1}^{K} s_k$, and $p_k = s_k/s$ is the fraction of the total data at node k.

FedAvg combines local stochastic gradient descent (SGD) at each node with a server that performs model averaging [2]. In each round t, the server sends the weights w^t of the global model, which might initially be empty, to the selected nodes K, where $K \subseteq N$. The fraction $C = N/K$ of selected nodes (called *sampling rate*) is set beforehand and remains the same throughout all rounds.

Each node k trains the global model locally and adjusts its weights through: $w_k^{t+1} = w^t - \eta \nabla F_k(w^t)$, where η is the learning rate and F_k the loss function. These weights are sent to the server, which averages them to compute the new global model:

$$w^{t+1} = \sum_{k=1}^{K} p_k \cdot w_k^{t+1}$$

Dynamic Sampling overcomes *FedAvg*'s static nature by dynamically adjusting the number of nodes selected in each round [9]. FedAvg defines the sampling rate C at the beginning and keeps it until the end. In contrast, Dynamic Sampling starts with a high sampling rate, that is, many nodes are selected, and gradually reduces it in each round to accelerate convergence. As the federated model matures, fewer nodes participate: This helps ensure convergence and saves communication resources. The sampling rate of the nodes is governed by a decay coefficient β. If C is the initial rate, $C_t = C/exp(\beta t)$, and thus the higher t, the fewer nodes are selected in each round.

Power of choice solutions aim to select nodes dynamically based on their local losses to achieve faster convergence [11]. The server seeks to collectively find the weights w that minimize the global loss function:

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

where $F_k(w)$ represents the loss function of node k.

These solutions dynamically select the nodes with the highest loss during training, following the intuition that the nodes with the highest loss can contribute more to the global model. Prioritizing them allows one to achieve a faster convergence speed. More specifically, when a server adopts a Power of choice solution, it performs three steps:

- Samples a candidate set of nodes choosing each node k with probability p_k equal to the fraction of the data at node k. This means that nodes with more data are likely to be selected.
- Sends the global weights w^t to the nodes to calculate their local losses $F_k(w^t)$.
- Forms the active set of nodes by choosing those with the highest $F_k(w^t)$.

Pow-d, also known as $\pi_{\text{pow-d}}$, is the base variant of this way of working and computes local losses considering the complete local data set D_k. Its two primary drawbacks are that it requires a preliminary phase in which each node k uses the entire local data set D_k, which requires time and resources, and all nodes communicate their local losses to the server in every round, which requires additional communication. **Cpow-d** [11] addresses the first limitation by estimating local losses by means of uniformly and randomly sampled mini batches of D_k. This improves efficiency by reducing computational requirements, although at the expense of a potential loss of representativeness. **Rpow-d** addresses both the computation and communication costs [11]. It skips the initial sampling, the nodes compute their cumulative average loss through local iterations, and the server uses these values to select the nodes. For nodes that have never been selected before, the most recent loss value is set to ∞.

B. Workload Optimization

Since nodes can be heterogeneous and can supply different resources, the workload each node can manage at each round must take it into account. To fine-tune the workload assigned to each node, we consider three parameters that influence the amount of computation required in each round: number of epochs E, batch size B, and fraction of samples r. We then consider four different optimization techniques: *Static Optimizer*, *Uniform Optimizer*, *Round Time Optimizer*, and *Equal Computation Time Optimizer*.

Static Optimizer is a naive strategy that statically configures epochs, batch sizes, and fraction of samples, treating all nodes uniformly. This basic solution, which does not consider node-specific features, is our baseline.

Uniform Optimizer assigns epochs, batch sizes, and fraction of samples to nodes by drawing from uniform distributions within specified ranges. Given a node k, the number of epochs E_k, the batch size B_k and the sample fractions r_k are drawn uniformly from a range of predefined values in $[val_{min}, val_{max})$. This way one can cope with workload heterogeneity, but does not adapt the workload to the specific needs of each node.

Round Time (RT) Optimizer configures batch sizes and sample fractions with predefined values. In contrast, for each node k, the number of epochs E_k is assigned proportionally to its computational power, measured in iterations per second (IPS). The node with the highest IPS is assigned the maximum epochs E_{max} from a range of values. This approach optimizes learning by selecting a higher computational load

TABLE I: Default parameters for the experiments.

Parameter	Symbol	Value
Number of nodes	N	100
Number of rounds	R	100
Number of epochs	E_k	4
Batch size	B_k	32
Fraction of samples	r_k	1.0

for faster nodes and can select the workload according to the computational capacities of the nodes.

Equal Computation Time (ECT) Optimizer configures epochs, batch sizes, and sample fractions based on a fixed desired computation time T and the IPS values IPS_k of each individual node k. Local iterations at the node are calculated as $I_k = T \cdot IPS_k$. The epochs at node k can be calculated as $E_k = I_k \cdot B/s_k \cdot r$, where s_k is the size of the data at node k, and the batch size B and sample fraction r are set to default values. Similarly, one could work on B and r, and consider the defaults for the other values. The choice of the variable parameter depends on the scenario: for example, varying sample fractions may imply inadequate updates for low IPS, while increasing batch sizes may stress limited memory, and excessive epochs can lead to over-fitting. This approach ensures equal computation time among nodes and addresses resource variations.

III. Assessment

This section describes the experiments we performed to evaluate the different strategies and the results we obtained. There are many ML tasks that can be carried out in an FL setting. This work focuses on image classification, since it is a widespread and popular activity, and the sensitive nature of images in certain contexts (e.g., healthcare) makes it particularly attractive for privacy-aware solutions [12]–[14]. Numerous open data sets are available [15], [16], and image classification tasks often involve non-IID data, that is, a realistic challenge for FL models [17], [18]. We considered a multi-layer perceptron (MLP) with two hidden layers of 64 and 30 neurons, respectively. Dropout is applied after the first hidden layer, and the input is represented by the flattened image. All experiments were run with the built-in Flower simulation engine, on a M2 Apple MacBook equipped with 64GB of RAM. Table I summarizes the default values of the FL parameters common to all experiments.

The experiments exploited the MNIST data set [16]. It comprises 28x28 pixel images of handwritten digits, with 60,000 images for training and 10,000 for testing, distributed across 10 classes corresponding to the digits from 0 to 9. We tested 100 nodes and used 100 rounds. We tried to simulate realistic data distributions to address data heterogeneity among nodes [4]–[6]. We adopted the method proposed by Hsu et al. [19] that uses a configurable Latent Dirichlet Allocation (LDA) distribution based on parameter $\mathrm{Dir}_K(\alpha)$ to construct heterogeneous data partitions among nodes, with parameter α that controls the degree of data heterogeneity: if $\alpha \to \infty$ all clients have identical distribution, and $\alpha \to 0$ each client holds samples from only one class. We experimented with α equal to 0.5, 1, 10, and 100, that is, from very unbalanced data to very stable ones[2]. Note that α influences both the number of samples in each partition and the balance of samples among classes.

A. Obtained Results

The evaluation considers the overall training loss and the final and maximum test accuracy of the aggregated models. For workload optimization, we also present two additional metrics to assess the improvement in the convergence speed of the global model: the maximum training time among the nodes T_{max} and the variance of training times $\mathcal{S}^2$. Both are averaged over the training rounds. To compute these last two metrics, we first estimate the number of iterations per node as: $I_k = (s_k \cdot r_k/B_k) \cdot E_k$. We estimate the execution time for each node as $T_k = I_k/IPS_k$ (recall that IPS_k describes the computational power of node k in terms of the number of iterations per second). Then, we compute the maximum training time among nodes:

$$T_{max} = \frac{1}{R}\sum_{t=1}^{R} max(T_k)$$

where R is the total number of rounds (in our case 100). Similarly, the variance of training times among nodes is as follows:

$$\mathcal{S}^2 = \frac{1}{R}\sum_{t=1}^{R} \sigma^2$$

where σ^2 is the variance between nodes per round:

$$\sigma^2 = \frac{1}{K}\sum_{k=1}^{K} (T_k - \mu)^2$$

B. Node Selection

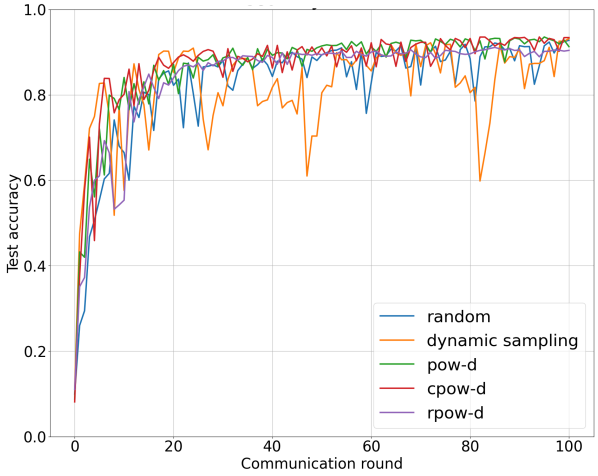
Table II shows the results obtained. *Power of choice* strategies (*pow-d*, *cpow-d*, *rpow-d*) demonstrated effectiveness, particularly in unbalanced datasets, outperforming *FedAvg* (*rand*) in terms of both convergence speed and stability. *Dynamic Sampling*, although rapidly converging, showed an instability in performance over time, especially in the last rounds of simulation. This can be attributed to the idea of initially involving a large number of nodes and gradually reducing them, which may affect the long-term stability of the model. *Dynamic Sampling* shows the best performance with large α's, that is, with balanced data distributions.

Figure 1 plots the evolution of test accuracy of the different strategies for the overall trained model. The two figures show the two extreme cases with $\alpha = 0.5$ and $\alpha = 100$, and we can easily see that the unbalanced data distribution implies fluctuating accuracy.

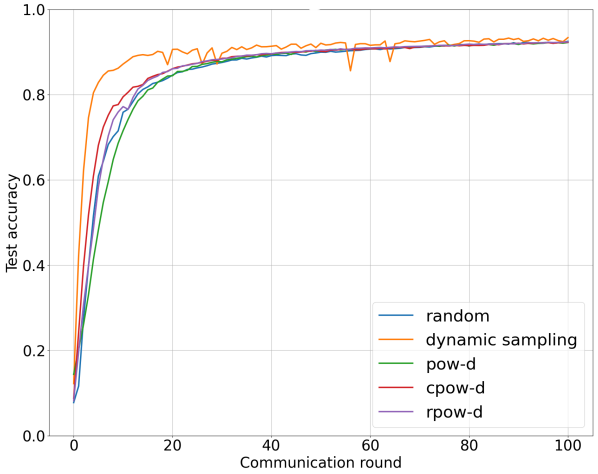
[2]We also carried out experiments with other values, but the four values presented here provide a fair summary of all results.

TABLE II: Node selection strategies (best results in bold).

α	Strategy	Max Acc	Final Acc	Training Loss
.5	*rand*	0.9276	0.9276	0.2485
	dyn samp	0.9315	0.9315	0.2350
	pow-d	**0.9354**	0.9126	0.2744
	cpow-d	0.9354	**0.9340**	**0.2153**
	rpow-d	0.9110	0.9039	0.3152
1	*rand*	0.9259	0.9226	0.2636
	dyn samp	0.9240	**0.8636**	0.4376
	pow-d	**0.9409**	0.9409	**0.1949**
	cpow-d	0.9395	0.9395	0.2072
	rpow-d	0.9208	0.9132	0.2825
10	*rand*	0.9218	0.9218	0.2696
	dyn samp	**0.9355**	0.9229	0.2788
	pow-d	0.9215	0.9211	0.2737
	cpow-d	0.9269	**0.9269**	**0.2562**
	rpow-d	0.9185	0.9179	0.2741
100	*rand*	0.9250	0.9250	0.2670
	dyn samp	**0.9337**	**0.9337**	**0.2178**
	pow-d	0.9222	0.9222	0.2708
	cpow-d	0.9225	0.9225	0.2762
	rpow-d	0.9236	0.9236	0.2760



(a) $\alpha = 0.5$.



(b) $\alpha = 100$.

Fig. 1: Accuracy evolution over time (dynamic node selection).

C. Workload Optimization

Table III presents the results obtained with the workload optimization strategies discussed above, together with *pow-d* for node selection, given its very good performance. *Static* and *RT* performed consistently well in terms of accuracy in the different settings and indicated their robustness. The former obtained the best accuracy performance in general, but the ability of the latter to optimize the workload to the different nodes makes it a closer and more efficient competitor. This is confirmed by comparing the time results expressed by T_{max} and $\mathcal{S}^2$: While *Static* obtained the worst results, *RT* shows comparable accuracy with significantly lower average maximum execution time and variance among nodes. The *ECT*, despite the excellent results with time metrics, suffers from lower accuracy, regardless of the degree of heterogeneity of the data. The random approach to the distribution of the workload of *uniform* resulted in poor overall performance and highlighted the need for strategies that take into account the characteristics of each node, such as computational power and the size of the data sets.

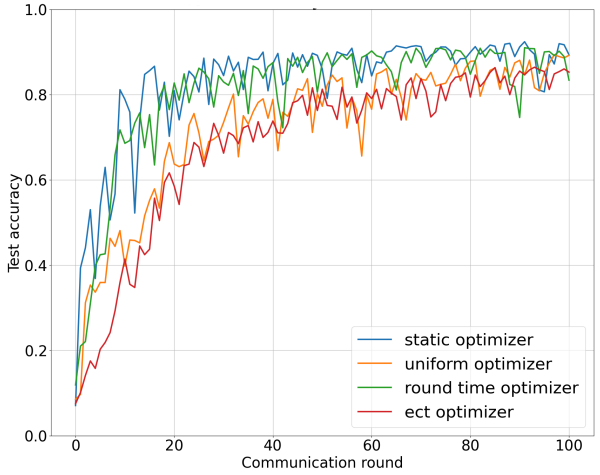
Figure 2 presents the evolution over time of the accuracy of all optimization strategies with the two extreme values for α. Although the general trend, fluctuation over more stability, is the same as before, the differences among the different optimizers are evident, with *RT* being consistently the best among the dynamic approaches.

Furthermore, Figure 3 compares the four strategies in terms of variance among nodes' $\mathcal{S}^2$ and mean maximum execution time T_{max} during the 100 training rounds. This figure once again highlights the excellent stability and convergence speed of *ECT*, and the similarities between *Uniform* and *RT* in keeping variance and maximum execution time low, while *Static* highlights significant spikes and fluctuations.

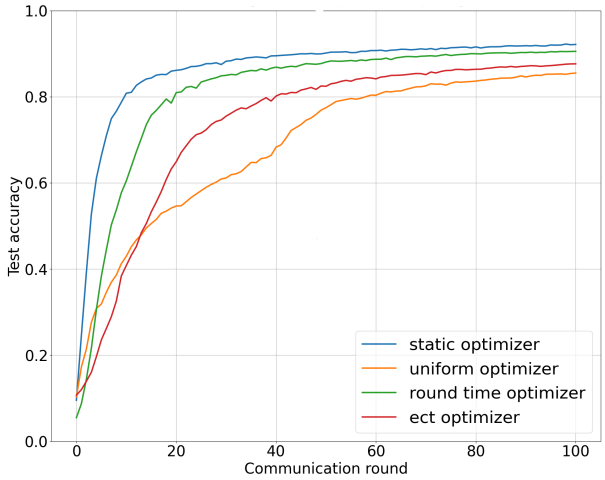
TABLE III: Workload optimization strategies (best results in bold).

α	Strat.	M. Acc	F. Acc	Tr. Loss	T_{max}	$\mathcal{S}^2$
.05	*static*	**0.9240**	**0.8932**	**0.3447**	34.57	171.21
	uniform	0.8932	0.8921	0.3978	5.98	4.53
	RT	0.9099	0.8338	0.4968	16.72	35.76
	ECT	0.8641	0.8526	0.5310	**1.85**	**0.44**
1	*static*	**0.9300**	**0.9215**	**0.2693**	28.08	75.76
	uniform	0.8902	0.8711	0.4336	7.51	6.52
	RT	0.9121	0.8881	0.3632	11.29	12.16
	ECT	0.8787	0.8766	0.4743	**1.85**	**0.30**
10	*static*	**0.9257**	**0.9247**	**0.2650**	13.51	8.93
	uniform	0.8712	0.8712	0.5023	4.22	1.61
	RT	0.9057	0.9049	0.3292	6.14	1.60
	ECT	0.8697	0.8697	0.4963	**1.85**	**0.01**
100	*static*	**0.9222**	**0.9212**	**0.2723**	12.65	6.82
	uniform	0.8549	0.8549	0.5519	3.55	1.02
	RT	0.9053	0.9053	0.3309	5.37	0.64
	ECT	0.8762	0.8762	0.4634	**1.87**	**0.01**

In general, the experiments highlight the importance of strategic node selection and workload distribution in FL to achieve more effective and stable model training.

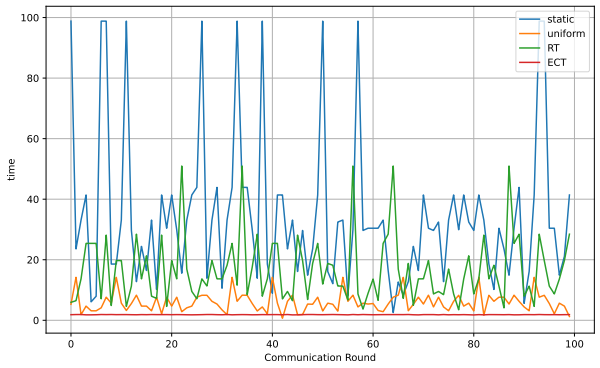


(a) $\alpha = 0.5$.

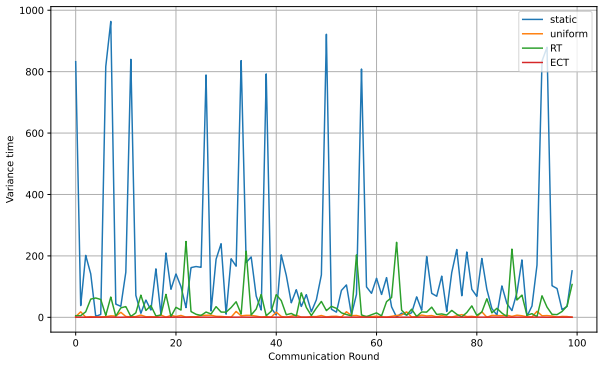


(b) $\alpha = 100$.

Fig. 2: Test accuracy evolution over time (workload optimization).



(a) Variance



(b) Max Time

Fig. 3: Maximum execution time and variance across nodes for workload optimizers.

D. Threats to Validity

Although FL is frequently used to train models on large amounts of data, our experiments employ a relatively small data set. This choice allowed us to develop a manageable experimental environment and to run experiments faster. The size of the data set we used is sufficient to generalize the results obtained, but our findings may be different with larger data sets. Another key challenge lies in reproducing the degree of heterogeneity. Although we relied on state-of-the-art techniques (i.e., [19]), we are aware that in a real-world context heterogeneity can manifest itself in more complex and unpredictable ways. Another factor to consider is the variation in computational power within the federation. Despite simulating the variability of computational resources among nodes, real ones may exhibit unpredictable decreases in available resources, with some stopping computation due to external factors.

Experiments in real settings are key to further validate the results presented in this paper, but we are convinced that these results provide a significant and better understanding of the different alternatives one must consider when conceiving an FL-based solution.

IV. Related Work

FL still has many unsolved challenges [20]. As for heterogeneity different surveys have already addressed the issue from different angles.

Zhang et al. [21] provide an extensive review of federated learning mechanisms, emphasizing the privacy-preserving aspect and the challenges associated with data distribution among heterogeneous networks. Although it covers many aspects, it does not provide any empirical comparison among alternative solutions. A more recent survey by Mora et al. [22] focuses on improving the generalization of federated learning models under the conditions of data heterogeneity. The work systematically categorizes and evaluates various state-of-the-art approaches and emphasizes methods to improve model generalization across diverse client datasets. Although this survey provides a very good analysis and comparisons of various techniques for managing data heterogeneity, its primary focus on generalization offers a complementary perspective to our work, which empirically evaluates node selection and workload optimization in detail. Another survey by Ma et al [23].

provides a systematic review of approaches to the problem of non-IID data, which is closely related to our exploration of heterogeneity management. This survey categorizes the solutions into four primary dimensions: data-based, model-based, algorithm-based, and framework-based. They focus mainly on the theoretical aspects and classification of existing methods; our work complements this by providing empirical evaluations and comparisons of some solutions. Finally, Lu et al. [24] present a comprehensive survey that includes various solutions for heterogeneity in federated learning. They address the effect of non-IID data on communication efficiency, model convergence, and achieved accuracy. Again, this work is more conceptual, while ours is a step forward and puts some solutions into practice.

V. Conclusions

This paper analyzes and compares dynamic node selection and resource-aware workload optimization strategies to (partially) address heterogeneity in FL. Our future work comprises the analysis of other strategies and dimensions to address heterogeneity, more experiments with a broader variety of tasks from different domains (e.g., natural language processing), and also experiments on a real edge infrastructure.

Appendix

Table IV lists the default values we used for the experiments.

TABLE IV: Default values used in the experiments.

Par.	**Sel. Strategy**	**Description**	**Value**
C	fedAvg, pow-d, cpow-d, rpow-d	Fraction of nodes/round	0.1
d	pow-d, cpow-d	Number of nodes, first phase	20
b	cpow-d	Mini-batch size, first phase	64
C_0	dyn. sampling	Initial fraction of nodes	0.2
β	dyn. Sampling	Sampling rate decay	0.1
Par.	**Optimizer**	**Description**	**Value**
E_{min}	uniform	Minimum number of epochs	1
E_{max}	uniform	Maximum number of epochs	5
B_{min}	uniform	Minimum batch size	32
B_{max}	uniform	Maximum batch size	128
r_{min}	uniform	Minimum fraction of samples	0.1
r_{max}	uniform	Maximum fraction of samples	1.0
E	ECT	Default number of epochs	2
B	ECT	Default batch size	32
r	ECT	Default fraction of samples	1.0
T	RT, ECT	Computation time for nodes	30
IPS_{mean}	RT, ECT	IPS mean value	100
IPS_{var}	RT, ECT	IPS variance value	50

Acknowledgments: This work has been supported by project COBOL (COmmunity-Based Organized Littering), funded by the MUR under the PRIN 2022 PNRR program (contract nr. P20224K9EK).

References

[1] P. Kairouz *et al.*, "Advances and open problems in federated learning," *Foundations and Trends® in Machine Learning*, vol. 14, no. 1–2, pp. 1–210, 2021.

[2] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*. PMLR, 2017, pp. 1273–1282.

[3] N. Rieke, J. Hancox, W. Li, F. Milletari, H. R. Roth, S. Albarqouni, S. Bakas, M. N. Galtier, B. A. Landman, K. Maier-Hein *et al.*, "The future of digital health with federated learning," *NPJ digital medicine*, vol. 3, no. 1, p. 119, 2020.

[4] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE signal processing magazine*, vol. 37, no. 3, pp. 50–60, 2020.

[5] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of fedavg on non-IID data," in *8th International Conference on Learning Representations*, 2020.

[6] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-IID data," *CoRR*, vol. abs/1806.00582, 2018. [Online]. Available: http://arxiv.org/abs/1806.00582

[7] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," *Proceedings of Machine learning and systems*, vol. 2, pp. 429–450, 2020.

[8] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konečnỳ, S. Mazzocchi, B. McMahan *et al.*, "Towards federated learning at scale: System design," *Proceedings of machine learning and systems*, vol. 1, pp. 374–388, 2019.

[9] S. Ji, W. Jiang, A. Walid, and X. Li, "Dynamic sampling and selective masking for communication-efficient federated learning," *IEEE Intelligent Systems*, vol. 37, no. 2, pp. 27–34, 2021.

[10] D. J. Beutel, T. Topal, A. Mathur, X. Qiu, T. Parcollet, and N. D. Lane, "Flower: A friendly federated learning research framework," *CoRR*, vol. abs/2007.14390, 2020. [Online]. Available: https://arxiv.org/abs/2007.14390

[11] Y. Jee Cho, J. Wang, and G. Joshi, "Towards understanding biased client selection in federated learning," in *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, vol. 151. PMLR, 28–30 Mar 2022, pp. 10 351–10 375.

[12] L. Li, N. Xie, and S. Yuan, "A federated learning framework for breast cancer histopathological image classification," *Electronics*, vol. 11, no. 22, 2022.

[13] B. Liu, B. Yan, Y. Zhou, Y. Yang, and Y. Zhang, "Experiments of federated learning for COVID-19 chest x-ray images," *CoRR*, vol. abs/2007.05592, 2020. [Online]. Available: https://arxiv.org/abs/2007.05592

[14] D. Ye, R. Yu, M. Pan, and Z. Han, "Federated learning in vehicular edge computing: A selective model aggregation approach," *IEEE Access*, vol. 8, pp. 23 920–23 935, 2020.

[15] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009, University of Toronto, ON, Canada.

[16] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov 1998.

[17] W. Hao, M. El-Khamy, J. Lee, J. Zhang, K. J. Liang, C. Chen, and L. C. Duke, "Towards fair federated learning with zero-shot data augmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 3310–3319.

[18] W. Huang, M. Ye, and B. Du, "Learn from others and be yourself in heterogeneous federated learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 10 143–10 153.

[19] T. H. Hsu, H. Qi, and M. Brown, "Measuring the effects of non-identical data distribution for federated visual classification," *CoRR*, vol. abs/1909.06335, 2019. [Online]. Available: http://arxiv.org/abs/1909.06335

[20] L. Baresi, G. Quattrocchi, and N. Rasi, "Open challenges in federated machine learning," *IEEE Internet Computing*, vol. 27, no. 2, pp. 20–27, 2023.

[21] C. Zhang, Y. Xie, H. Bai, B. Yu, W. Li, and Y. Gao, "A survey on federated learning," *Knowledge-Based Systems*, vol. 216, p. 106775, 2021.

[22] A. Mora, A. Bujari, and P. Bellavista, "Enhancing generalization in federated learning with heterogeneous data: A comparative literature review," *Future Generation Computer Systems*, 2024.

[23] X. Ma, J. Zhu, Z. Lin, S. Chen, and Y. Qin, "A state-of-the-art survey on solving non-iid data in federated learning," *Future Generation Computer Systems*, vol. 135, pp. 244–258, 2022.

[24] Z. Lu, H. Pan, Y. Dai, X. Si, and Y. Zhang, "Federated learning with non-iid data: A survey," *IEEE Internet of Things Journal*, 2024.