Entropy-Driven Stochastic Policy for Fast Federated Learning in Beyond 5G Edge-RAN

Brahim Aamer[1], Hatim Chergui[2], Mustapha Benjillali[1], and Christos Verikoukis[2]
[1]INPT, Rabat, Morocco
[2]CTTC, Barcelona, Spain
Emails: aamer.brahim@gmail.com, benjillali@ieee.org, {hatim.chergui, cveri}@cttc.es

***Abstract*—Scalability and sustainability are the corner stones to unleash the potential of beyond fifth-generation (B5G) ultra-dense networks that are expected to handle massive and heterogeneous services. This implies that the transport of the underlying raw monitoring data should be minimized across the network, and urges to bring the analysis functions closer to the data collection points. While federated learning (FL) is an efficient tool to implement such a decentralized strategy, real networks are generally characterized by time- and space-varying users distributions, traffic profiles, and channel conditions. This makes the data collected across different points non independent and identically distributed (non-IID), which is challenging for FL tasks. To cope with this issue, we first introduce a new *a priori* metric that we call *dataset entropy*, whose role is to capture the distribution, the quantity of information, the unbalanced structure and the "non-IIDness" of a dataset independently of the models. This entropy is calculated using a clustering scheme based on a similarity matrix defined over both the features and the supervised output spaces, and is targeting classification as well as regression tasks. The FL aggregation server then uses the reported dataset entropies to devise i) an entropy-based federated averaging scheme, and ii) a stochastic participant selection policy to significantly stabilize the training, minimize the convergence time, and reduce the corresponding computation cost. Numerical results are provided to illustrate all these advantages.**

I. Introduction

With the proliferation of beyond 5G (B5G) use cases, wireless networks need to manage a massive number of simultaneous and heterogeneous services, which makes the classical centralized monitoring, analysis, and control impractical, as they usually represent a single point of failure and suffer from large overhead and delay. Alternatively, decentralized service processing guarantees scalability, low data exchange and, therefore, more security. In this regard, distributed artificial intelligence (AI) approaches, and in particular FL schemes, can play a pivotal role in unleashing the full potential of scattered monitoring data across the network and leveraging the computing power brought by both the edge cloud and the fog devices, while reducing the computational costs and enabling fast local analysis and decision. Nonetheless, FL capability is often limited by the convergence delay that spans dozens of rounds due to several conceptual and operational issues that are reviewed in the sequel.

A. Related Work

In [3], the authors have introduced the federated averaging (FedAvg) algorithm that synchronously aggregates the parameters, and is thus susceptible to the straggler effect, i.e., each training round only progresses as fast as the slowest edge device since the FL server waits for all devices to complete local training before the global aggregation can take place. Alternatively, the asynchronous model in [4] has been proposed to improve the scalability and efficiency of FL. For asynchronous FL, the server updates the global model whenever it receives a local update which grants more robustness against participants joining halfway during a training round, as well as when the federation involves participating devices with heterogeneous processing capabilities. However, the model convergence is found to be significantly delayed when data is non independent and identically distributed (non-IID) and unbalanced [5]. To cope with this issue, it has been proposed to distribute the available data publicly to participants. However, such a dataset may not always exist, or the participants may refuse to download them for security reasons. Thus, an alternative solution is to construct an approximately IID dataset using inputs from a limited number of privacy insensitive participants [6]. In the Hybrid-FL protocol, the server asks random participants if they allow their data to be uploaded. During the participant selection phase, apart from selecting participants based on computing capabilities, participants are selected such that their uploaded data can form an approximately IID dataset in the server, i.e., the amount of collected data in each class has close values. Thereafter, the server trains a model on the collected IID dataset, and merges this model with the global model trained by the participants. Nevertheless, requests for data sharing are not in line with the original intent of FL. As an improvement, the authors in [7] have proposed the FedAsync algorithm in which newly received local updates are adaptively weighted according to staleness, that is defined as the difference between the current epoch and the iteration to which the received update belongs to. For example, a stale update from a straggler is outdated since it should have been received in previous training rounds. As such, it is given a smaller weight. In addition, the authors prove the convergence guarantee for a restricted family of non-convex problems. However, the current hyperparameters of the FedAsync algorithm still have to be tuned to ensure convergence in different settings. Hence, the algorithm is still unable to generalize to suit the dynamic computation constraints of heterogeneous devices. Given this uncertainty

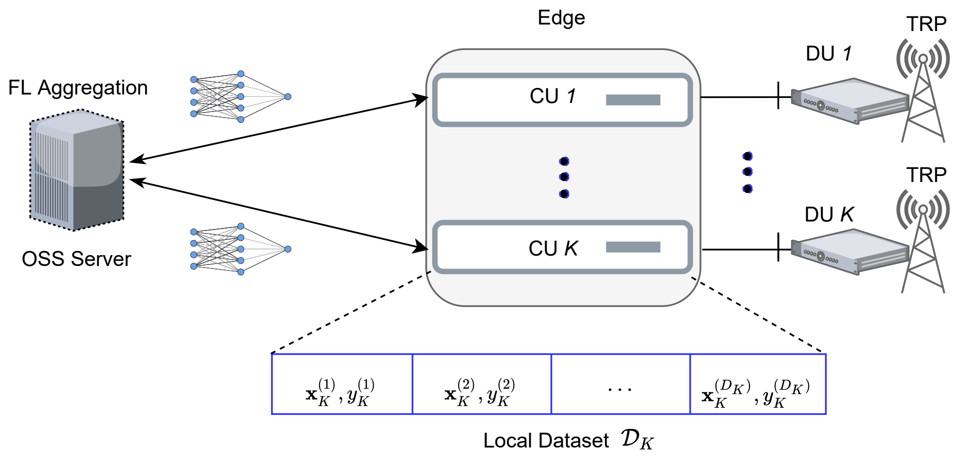

Edge
FL Aggregation
CU 1
DU 1
TRP
OSS Server
CU K
DU K
TRP
$\mathbf{x}_K^{(1)}, y_K^{(1)}$
$\mathbf{x}_K^{(2)}, y_K^{(2)}$
$\cdots$
$\mathbf{x}_K^{(D_K)}, y_K^{(D_K)}$
Local Dataset $\mathcal{D}_K$

Fig. 1. Network architecture

surrounding the reliability of asynchronous FL, synchronous FL remains the most commonly used approach [8]. In this context, it has been confirmed that the correlation between the model parameters of different clients is increasing as the training progresses, which implies that aggregating parameters directly by averaging may not be a reasonable approach in general [9]. Finally, a fair resource federated learning approach has been studied recently in [10], which introduces a weighted averaging that gives higher weights to devices with the worst performance (i.e., the largest loss) to let them dominate the objective, and thereby impose more uniformity to the training accuracy.

B. *Contributions*

In this paper, our contribution is two-fold.

- We first introduce the concept of the *entropy* of a dataset in both classification and regression tasks, where we jointly consider the features and supervised outputs to characterize the distribution of its samples and the underlying quantity of information based on a custom spectral clustering strategy. This generalized entropy captures the diversity of a dataset as well as its unbalanced structure and non-IIDness.
- By leveraging the introduced entropy as an *a priori* information, we devise two novel FL strategies, namely, i) Entropy-weighted federated aggregation which involves all the CUs in the FL training task while prioritizing the most balanced and uncorrelated datasets (i.e., those maximizing the entropy) and ii) Entropy-driven stochastic policy for selecting only a subset of CUs to take part in the FL task. This consists on sampling, at each FL round, the active CUs according to an entropy-based probability distribution, which dramatically reduces the convergence stability and time, as well as the underlying resource consumption by avoiding concurrent training by all CUs at each round.

II. Network Description and Data Collection

A. *Edge-RAN*

As depicted in Fig. 1, the considered network corresponds to a beyond 5G edge-RAN under the central unit (CU)/distributed unit (DU) functional split, where each transmission/reception point (TRP) is co-located with its DU, while all CUs are hosted in an edge cloud where they run as virtual network functions (VNFs). Each CU $k\,(k = 1, \ldots, K)$ performs RAN key performance indicators (KPIs) data collection to build its local dataset $\mathcal{D}_k = \{\mathbf{x}_k^{(i)}, y_k^{(i)}\}_{i=1}^{D_k}$ of size D_k, where $\mathbf{x}_k^{(i)}$ stands for the input features vector while $y_k^{(i)}$ represents the corresponding output. Given that this dataset is generally non-exhaustive to train accurate analytical models, the CU takes part in a federated learning task wherein an operational subsystem (OSS) server—located at the core cloud—plays the role of a model aggregator.

B. *Data Collection*

Table I shows the features and the supervised output of the local datasets, which have been collected from a live LTE-Advanced (LTE-A) RAN with a granularity of 1 hour. The considered TRPs cover areas with different traffic profiles—both in space and time—that tightly depend on the heterogeneous users distribution and behavior in each context (e.g., residential zones, business zones, entertainment events, ...). On the other hand, the radio KPIs are correlated with the time-varying channel conditions. These realistic datasets are therefore non-IID, which is more challenging for FL algorithms as studied in [11].

III. Proposed Entropy-Based Federated Learning

To tackle the FL convergence in practical non-IID setups, we seek an objective and compressed metric capturing both the distribution of a dataset and its quantity of information, while not depending on the local models. In this regard, we introduce the notion of *dataset entropy* that is a sufficient statistic to characterize the unbalanced structure of a dataset, as well as its independence from other datasets. Specifically, the entropy

TABLE I
DATASET FEATURES AND OUTPUT

Feature	Description
`Cell Throughput`	Cell Downlink Average Throughput
`User Throughput`	User Downlink Average Throughput
`BLER`	Average Block Error Rate
`# Users`	Downlink Average Active Users
`MIMO Rank`	Average MIMO Rank
`DL PRB`	Downlink PRB Usage Percentage
`TA`	Average Timing Advance
`CQI`	Average Channel Quality Index
`QPSK`	Percentage of QPSK modulation usage

Output	Description
`Traffic`	Traffic Volume (Output)

is maximized under a uniform distribution with low probability mass function (PMF). By relying on the *a priori* entropies of all CUs, the aggregation server can implement novel CUs selection and models combining schemes to accelerate and stabilize the FL convergence.

A. Dataset Entropy

Since we are targeting a generalized definition of the entropy, the labels of a classification dataset are not reliable to reflect the distribution of data since it does not apply to regression tasks where the supervised output is continuous, and it omits the effect of the input features. In particular, samples with different feature values but presenting approximately similar outputs are not providing the same information and might not necessarily fit in the same group of data. Therefore, in order to accurately discern the samples, we consider a joint approach where both the features and the supervised output are used. To that end, each CU uses a clustering algorithm that operates on the so-called *similarity matrix* $\mathbf{S}_k$ whose entries measure the logical correlations between the dataset samples vectors including both the features and the supervised output, i.e.,

$$\tilde{\mathbf{x}}_k^{(i)} = \left[\mathbf{x}_k^{(i)} \; y_k^{(i)}\right]. \tag{1}$$

This matrix is built using a radial basis function (RBF) kernel with parameter σ. As such, the (i,j)-th matrix element is given by

$$s_{i,j}^{(k)} = \exp\left(-\frac{d\left(\tilde{\mathbf{x}}_k^{(i)}, \tilde{\mathbf{x}}_k^{(j)}\right)}{\sigma^2}\right), \tag{2}$$

where d stands for the pairwise logical distance between samples' vectors i and j. Since basic clustering algorithms usually require the target number of clusters as an input, we resort to the well-established self-tuning spectral clustering (STSC) technique that uses the eigenvalues and eigenvectors of the similarity matrix. This enables to automatically cluster a dataset into an appropriate number of clusters that minimizes a custom cost function defined in terms of the coefficients of a rotated and normalized version of matrix $\mathbf{S}_k$ [12]. Let us assume that for CU k, the clustering yields n_k clusters $\mathcal{C}_{k,1},...,\mathcal{C}_{k,n_k}$ with probabilities $\Pr(\mathcal{C}_{k,1}),\ldots,\Pr(\mathcal{C}_{k,n_k})$ over dataset $\mathcal{D}_k$. The corresponding entropy is then defined as

$$\varepsilon_k = -\sum_{p=1}^{n_k} \Pr(\mathcal{C}_{k,p}) \log\{\Pr(\mathcal{C}_{k,p})\}. \tag{3}$$

By letting the CUs report their dataset entropies $\{\varepsilon_k\}_{k=1}^K$ to the aggregation server before starting the training, it becomes possible to devise advanced entropy-driven FL strategies that prioritize the CUs with high entropy datasets.

B. Entropy-Driven FL Combining

In this strategy, the aggregation server directly uses the entropies to perform a weighted averaging of all CUs local models at each round t, i.e.,

$$\mathbf{W}^{(t+1)} = \sum_{k=1}^{K} \frac{\varepsilon_k}{\sum_{p=1}^{K}\varepsilon_p}\mathbf{W}_k^{(t)}, \tag{4}$$

which allows the CUs with high entropies to dominate and orient the FL training, although this requires the participation of all CUs.

C. Entropy-Driven Stochastic FL Policy

To optimize the federated learning computation time as well as the underlying resource consumption, we aim at selecting only a number of active CUs in each FL round. In this respect, we introduce an entropy-driven stochastic CU selection policy wherein the aggregation server first generates a probability distribution over all the CUs using their received entropies. This is achieved by a direct `softmax` activation layer, i.e.,

$$\pi_k = \frac{\exp\{\varepsilon_k\}}{\sum_{p=1}^{K}\exp\{\varepsilon_p\}}. \tag{5}$$

Next, at each FL round t, as illustrated in Fig. 2, the server selects a subset of $m < K$ CUs to participate in the training by sampling the non-uniform CUs set with probabilities $\{\pi_1,\ldots,\pi_K\}$, i.e.,

$$\mathrm{CU}_{k_1}^{(t)},\ldots,\mathrm{CU}_{k_m}^{(t)} \sim \{\pi_1,\ldots,\pi_K \mid \mathrm{CU}_1,\ldots,\mathrm{CU}_K\}, \tag{6}$$

which ensures that, by the convergence round, the CUs would have stochastically taken part in the FL task according to the initial probability distribution, while avoiding the concurrent training by all CUs at each round. In this case, the model averaging at round t is performed as

$$\mathbf{W}^{(t+1)} = \sum_{k\in\{k_1,\ldots,k_m\}} \frac{D_k}{D}\mathbf{W}_k^{(t)}. \tag{7}$$

Where D is the total samples over all CUs datasets. This entropy-driven stochastic policy is summarized in Algorithm 1, where $\mathcal{L}(\cdot,\cdot)$ stands for the mean square error (MSE) loss function, and $\mathbf{b}$ is the bias, while the rest of FL setting parameters is provided in Table II.

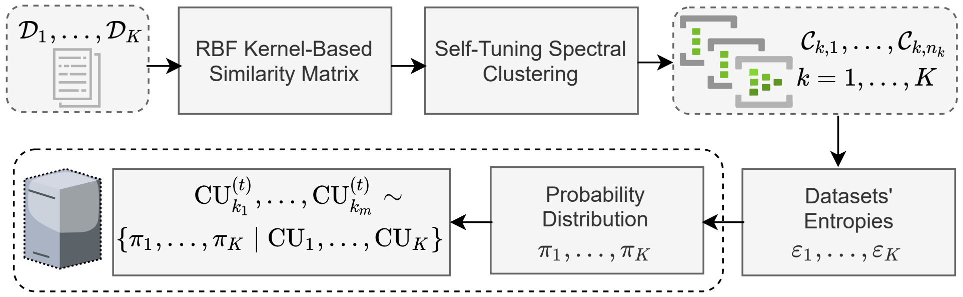

$\mathcal{D}_1,\ldots,\mathcal{D}_K$
RBF Kernel-Based Similarity Matrix
Self-Tuning Spectral Clustering
$\mathcal{C}_{k,1},\ldots,\mathcal{C}_{k,n_k}$ $k=1,\ldots,K$
Datasets' Entropies $\varepsilon_1,\ldots,\varepsilon_K$
Probability Distribution $\pi_1,\ldots,\pi_K$
$\mathrm{CU}^{(t)}_{k_1},\ldots,\mathrm{CU}^{(t)}_{k_m}\sim\{\pi_1,\ldots,\pi_K\mid\mathrm{CU}_1,\ldots,\mathrm{CU}_K\}$

Fig. 2. Entropy-Driven Stochastic Federated Learning Policy.

Algorithm 1: Entropy-Driven Stochastic Federated Learning Policy.

```
Input: $K$, $m$, $\eta$, $T$, $L$. # See Table II
parallel for $k=1,\ldots,K$ do
# Calculate RBF-based Similarity Matrix entries
$s^{(k)}_{i,j}=\exp\left(-\frac{d\left(\tilde{\mathbf{x}}^{(i)}_k,\tilde{\mathbf{x}}^{(j)}_k\right)}{\sigma^2}\right),\ 1\le i,j\le D_k$
# Clustering
CU $k$ clusters $\mathcal{D}_k$ based on the eigenvetors of matrix $\mathbf{S}_k$
return $n_k$, $\Pr\left(\mathcal{C}_{k,1}\right),\ldots,\Pr\left(\mathcal{C}_{k,n_k}\right)$
# Calculate Dataset Entropy
$\varepsilon_k=-\sum_{p=1}^{n_k}\Pr\left(\mathcal{C}_{k,p}\right)\log\Pr\left(\mathcal{C}_{k,p}\right)$
CU $k$ reports $\varepsilon_k$ to the aggregation server
end parallel for
# Federated Learning
# Server Generates Probability Distribution
for $k=1,\ldots,K$ do
    $\pi_k=\frac{\exp\{\varepsilon_k\}}{\sum_{l=1}^{K}\exp\{\varepsilon_l\}},\ k=1,\ldots,K$
end
Server initializes $\mathbf{W}^{(0)}$ with random Gaussian weights
for $t=0,\ldots,T-1$ do
    # Server Samples the $m$ CUs
    $\mathrm{CU}^{(t)}_{k_1},\ldots,\mathrm{CU}^{(t)}_{k_m}\sim\{\pi_1,\ldots,\pi_K\mid\mathrm{CU}_1,\ldots,\mathrm{CU}_K\}$
    Server broadcasts $\mathbf{W}^{(0)}$ to the $m$ selected CUs
    parallel for $k\in\{k_1,\ldots,k_m\}$ do
    # Local epochs
    for $l=0,\ldots,L-1$ do
        $\mathbf{W}_{k,l}=\mathbf{W}_{k,l-1}-\eta\nabla\mathcal{L}\left(\mathbf{W},\mathbf{b}\right)$
    end
    return $\mathbf{W}^{(t)}_k=\mathbf{W}_{k,L-1}$
    Each local CU $k$ sends $\mathbf{W}^{(t)}_k$ to the aggregation server.
    end parallel for
    # Server Aggregation
    return $\mathbf{W}^{(t+1)}=\sum_{k\in\{k_1,\ldots,k_m\}}\frac{D_k}{D}\mathbf{W}^{(t)}_k$
    Broadcasts $\mathbf{W}^{(t+1)}$ to all $K$ CUs.
end
```

TABLE II
FL SETTINGS

Parameter	Description	Value
T	Number of rounds	20
L	Number of epochs	50
K	Number of CUs	6
m	Number of selected CUs	3
D_k	Local dataset size	100
η	Learning rate	0.001
σ	Kernel parameter	1.0

IV. Numerical Results

A. Settings and Baselines

The structure of the global model weights matrix $\mathbf{W}$ has been defined by the server to satisfy the findings of [13], where the authors have estimated the required number Q of neurons per layer based on the number H of hidden layers, the dataset sizes D_k, and the number of features F as

$$Q=\frac{F+\sqrt{\max_{k=1,\ldots,K}D_k}}{H}, \tag{8}$$

which is confirmed via Fig. 3-4, where the best setting of the DNN model neurons turns out to be $Q=4$ for $H=3$. As a benckmark, the performance of our proposed approaches is compared with LossFedAvg [11] and FedAvg [3]. FL settings are listed on Table. II, where FL system consists of $K=6$ DUs running local DNN with a learning rate $\eta=0.001$ for $T=20$ rounds.

B. Numerical Results Analysis

Figures 5a and 5b illustrate the gains achieved by the entropy-weighted approach compared to the baseline FedAvg and LossFedAvg. The comparison is done for both balanced and unbalanced non IID datasets. As showcased in Table III, the entropy metric varies in balanced datasets, since the clustering technique takes into account the correlation between features as well as the supervised output. In the unbalanced scenario, the entropy difference between CUs is even clearer and demonstrates also that datasets with smaller size can sometimes yield more clusters compared to larger datasets, which further corroborates the role of the introduced entropy metric in characterizing a dataset efficiently.

A slightly lower losses are met with the entropy-weighted approach rather than the entropy stochastic policy, but both methods have the same convergence trend. In Fig. 5a and Fig. 5b both entropy-based FL converge faster than FedAvg and LossFedAvg. Knowing how critical is the bandwidth occupation for FL exchanges, and how the CUs local model training is power consuming, especially in B5G mobile systems, our introduced entropy stochastic policy shows good results. This aspect becomes more critical if the FL result is an input for fast decision-making algorithms such as network slicing orchestration or resources scheduling.

Better than FedAvg and LossFedAvg, the entropy stochastic policy convergence trend is oscillating around entropy-weighted

TABLE III
RESULTS: DATASETS CLUSTERING

CU number	Balanced			Unbalanced		
	Nb samples	Nb clusters	Entropy	Nb samples	Nb clusters	Entropy
1	100	2	0.692	100	2	0.692
2	100	2	0.592	70	3	1.026
3	100	2	0.676	90	2	0.515
4	100	3	0.998	80	4	1.238
5	100	3	1.051	50	3	1.068
6	100	2	0.676	60	2	0.690

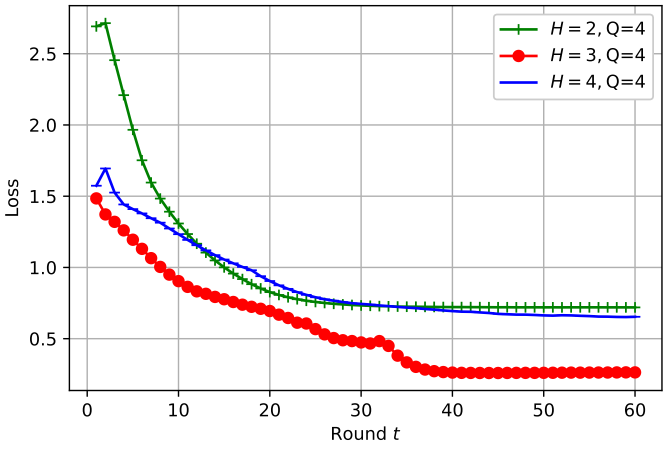


Fig. 3. Different number of layers configurations comparison.

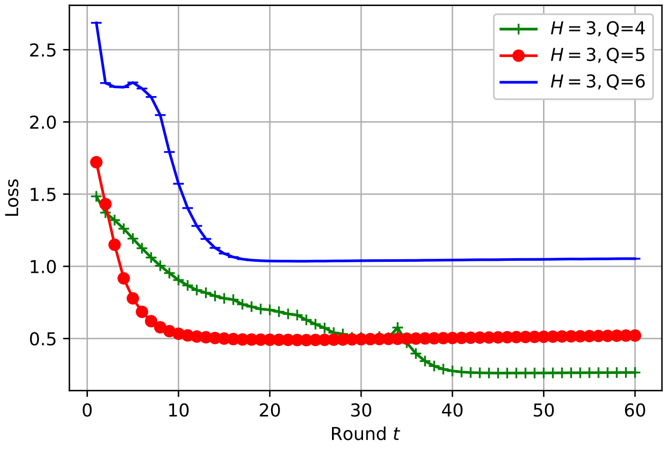
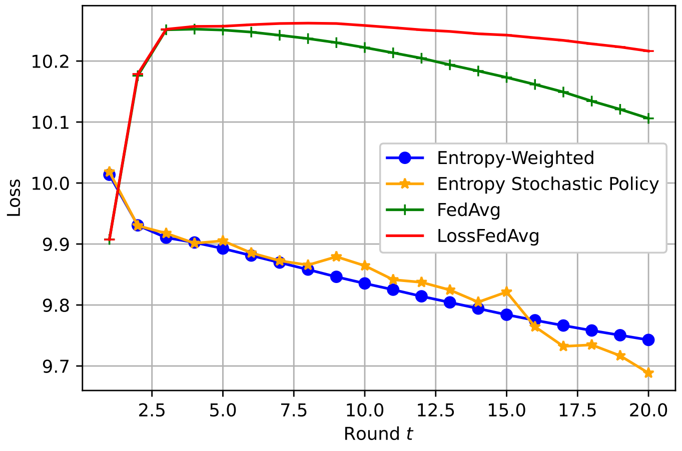
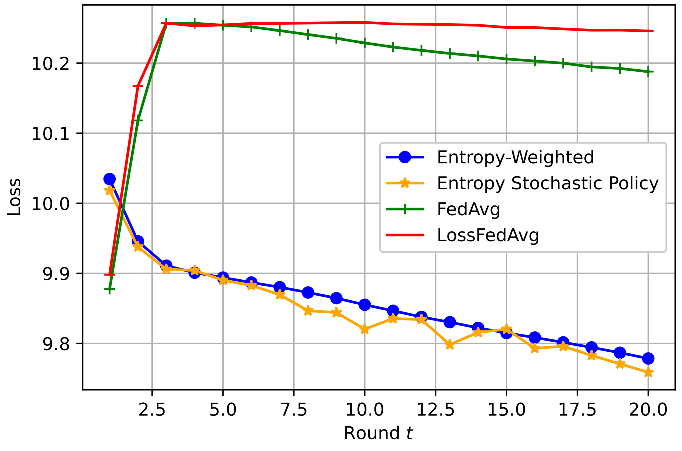


Fig. 4. Different number of neurons per layer configurations comparison.



(a) Balanced datasets



(b) Unbalanced datasets

Fig. 5. FL training loss vs. number of rounds.

as in Fig. 5a and Fig. 5b. Another important achievement with the entropy stochastic policy is the reduction of the required time for a given number of rounds and exchanges between the OSS server and the CUs towards convergence, as shown in Fig. 6, wherein the convergence time difference between the entropy-weighted approach and the entropy stochastic policy is exponentially growing with the number of FL rounds. Note that the corresponding wall-clock time performance is tightly dependent on the computation capabilities of both the OSS server and the CUs, but it shows that the stochastic policy FL minimizes the computation burden by selecting only a subset of CUs to take part in the training according to their *prior* entropy measure. More results can be generated for different values of K and m.

V. CONCLUSION

In this paper, we have introduced a novel *a priori* metric termed *dataset entropy* to characterize the distribution, the

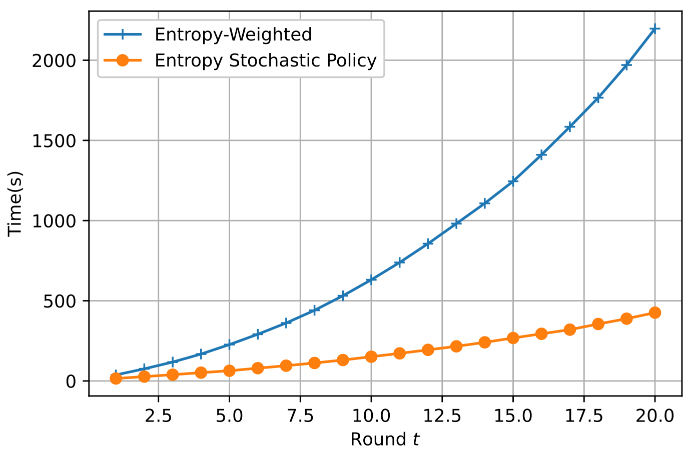


Fig. 6. Convergence time of entropy-weighted vs. entropy stochastic policy.

quantity of information, the unbalanced structure and the "non-IIDness" of a dataset independently of the models. This entropy is calculated via a generalized clustering strategy that relies on a custom similarity matrix defined over both the features and the supervised output spaces, and supporting both classification and regression tasks. The entropy metric has been then adopted to develop i) an entropy-based federated averaging scheme, and ii) a stochastic CU selection policy to significantly stabilize the training, minimize the convergence time, and reduce the corresponding computation cost. Numerical results have been provided to corroborate these findings. In particular, the convergence time difference between Entropy-Weighted and Entropy Stochastic Policy schemes is exponentially growing with the number of FL rounds.

References

[1] 3GPP, "Evolved packet system (EPS) mobility management entity (MME) and serving GPRS support node (SGSN) related interfaces based on Diameter protocol," TS 29.272 v15.4.0, Jun. 2018.

[2] ETSI GS ZSM 002, "Zero-touch network and service management (ZSM)," *Reference Architecture,* Aug. 2019.

[3] H.-B. McMahan *et al.,* "Communication-efficient learning of deep networks from decentralized data", in *the 20th International Conference on Artificial Intelligence and Statistics (AISTATS'17).*

[4] M. R. Sprague *et al.,* "Asynchronous federated learning for geospatial applications," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases,* Springer, 2018, pp. 21-28.

[5] Y. Zhao *et al.,* "Federated learning with non-IID data," [Online]. Available: arxiv.org/abs/1806.00582, 2018.

[6] N. Yoshida *et al.,* "Hybrid-FL: Cooperative learning mechanism using non-IID data in wireless networks," [Online]. Available: arxiv.org/abs/1905.07210, 2019.

[7] C. Xie *et al.,* "Asynchronous federated optimization," [Online]. Available: arxiv.org/abs/1903.03934, 2019.

[8] K. Bonawitz *et al.,* "Towards federated learning at scale: System design," [Online]. Available: arxiv.org/abs/1902.01046, 2019.

[9] P. Xiao *et al.,* "Averaging is probably not the optimum way of aggregating parameters in federated learning," *MDPI Entropy Journal,* vol. 22, no. 3, 2020.

[10] M. Tian Li *et al.,* "Fair resource allocation in federated learning," in *ICLR 2020.*

[11] L. Qinbin *et al.,* "Federated learning on Non-IID data silos: An experimental study," in *Computer Science,* Feb. 2021.

[12] L. Zelnik-Manor *et al.,* "Self-tuning spectral clustering," in *the 17th International Conference on Neural Information Processing Systems (NIPS'04),* pp. 1601-1608, 2004.

[13] L. J. Ke *et al.,* "Empirical analysis of optimal hidden neurons in neural network modeling for stock prediction," in *IEEE Pacific-Asia Workshop on Computational Intelligence and Industrial Application,* 2008.

[14] M. Chen *et al.,* "A joint learning and communications framework for federated learning over wireless networks," in *IEEE Transactions on Wireless Communications,* vol. 20, no. 1, pp. 269-283, Jan. 2021.

[15] WOOL, "STSC," 2018. [Online] Available: github.com/wOOL/STSC

[16] HIPS Group, "Autograd," 2018. [Online] Available: github.com/HIPS/autograd

[17] J. Townsend *et al.,* "Pymanopt: A Python toolbox for optimization on manifolds using automatic differentiation," in *Journal of Machine Learning Research,* vol. 17, no. 137, pp. 1-5, Aug. 2016. [Online] Available: pymanopt.github.io/

[18] F. Pedregosa *et al.,* "Scikit-learn: Machine learning in Python," in *Journal of Machine Learning Research,* vol. 12, pp. 2825-2830, Oct. 2011.