Received 12 June 2023, accepted 18 July 2023, date of publication 25 July 2023, date of current version 2 August 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3298704

RESEARCH ARTICLE

Entropy to Mitigate Non-IID Data Problem on Federated Learning for the Edge Intelligence Environment

FERNANDA C. ORLANDI[1], JULIO C. S. DOS ANJOS[2], VALDERI R. Q. LEITHARDT[3,4], (Senior Member, IEEE), JUAN FRANCISCO DE PAZ SANTANA[5], AND CLAUDIO F. R. GEYER[1]

[1]Federal University of Rio Grande do Sul, Institute of Informatics, UFRGS/PPGC, Porto Alegre, Rio Grande do Sul 91501-970, Brazil
[2]Federal University of Ceará, Graduate Program in Teleinformatics Engineering (PPGETI/UFC), Center of Technology, Campus of Pici, Fortaleza, Ceará 60455-970, Brazil
[3]VALORIZA-Polytechnic Institute of Portalegre, 7300-555 Portalegre, Portugal
[4]Instituto Superior de Engenharia de Lisboa (ISEL), Instituto Politécnico de Lisboa, 1959-007 Lisbon, Portugal
[5]Expert Systems and Applications Laboratory (ESALAB), Faculty of Science, University of Salamanca, 37008 Salamanca, Spain

Corresponding author: Fernanda C. Orlandi (fcorlandi@inf.ufrgs.br)

This work was partly sponsored by CAPES (Finance Code 001) and by the Spanish Agencia Estatal de Investigación through the Project Monitoring and Tracking Systems for the Improvement of Intelligent Mobility and Behavior Analysis (SiMoMIAC) under Grant PID2019-108883RB-C21/AEI/10.13039/501100011033.

ABSTRACT Machine Learning (ML) algorithms process input data making it possible to recognize and extract patterns from a large data volume. Likewise, Internet of Things (IoT) devices provide knowledge in a Federated Learning (FL) environment, sharing parameters without compromising their raw data. However, FL suffers from non-independent and identically distributed (non-iid) data, which means it is heterogeneous data and has biased input data, such as in smartphone data sources. This bias causes low convergence for ML algorithms and high energy and bandwidth consumption. This work proposes a method that mitigates non-iid data through a FedAvg-BE algorithm that provides Federated Learning with the border entropy evaluation to select quality data from each device by cross-device in a non-iid data environment. Extensive experiments were performed using publicly available datasets to train deep neural networks. The experiment result evaluation demonstrates that execution time saves up to 22% for the MNIST dataset and 26% for the CIFAR-10 dataset, with the proposed model in Federated Learning settings. The results demonstrate the feasibility of the proposed model to mitigate non-iid data impact.

INDEX TERMS Big data, collaborative learning, federated learning, machine learning, non-IID.

I. INTRODUCTION

Collaborative Learning is inspired by an educational approach to teaching and learning that involves groups of students working together to solve a problem, complete a task or create a product. This concept is expanded to Machine Learning (ML) [1], using agents that collect information from the environment to learn patterns according to some model established in a parallel and distributed architecture.

ML is one of the most important areas of Artificial Intelligence. It makes it possible to learn a model or a pattern of behavior for a given machine to perform tasks. An ML algorithm allows you to process input data with appropriate patterns to generate output data. Thus, it is possible to recognize and extract patterns from a large volume of data (Big Data) to build a learning model [2], serving for predictive modeling and decision-making. In a distributed architecture

The associate editor coordinating the review of this manuscript and approving it for publication was Amin Zehtabian.

it is possible to use consensus mechanisms to manage data consistency [3].

As is widely discussed in the literature when talking about Big Data, data production grows exponentially and quickly daily due to various sources and new emerging technologies [4]. There are smart devices as the main component of data sources, generating a large data volume. Some of these data sources have spatio-temporal characteristics, such as urban mobility, with a high degree of randomness [5]. In ML and data analysis, having homogeneous data independent and identically distributed (iid) with the same probability distribution and independent of each other is one of the common assumptions for hypothesis testing. However, objects, values, attributes, and everyday life are heterogeneous, non-independent, and identically distributed (non-iid) [6]. In practice, the training data may be insufficient due to recording errors or limited measurement conditions [7]. Non-iid data have a high degree of uncertainty, and the concept of entropy serves to measure this disorder in the data [8].

The initial hypothesis is that non-iid data is related to entropy. Using entropy computation to select quality data from heterogeneous data with lower entropy in Collaborative Learning (CL) models allows for a computation reduction time without substantial effect on accuracy. The exchange of information between devices can occur in a federated environment, which provides privacy for sensitive data such as medical and financial data [9]. The Federated Learning (FL) [10] environment provides raw data analysis without sharing it with other partners.

This paper proposes to evaluate the influence of entropy on data quality in IoT devices to reduce the impact of non-iid data, which normally have a problem of imbalance class distributions, in Federated Learning algorithms, using ML concepts and tools in federated distributed system environments. In the literature, some scientific articles present studies on those FL models to identify types of algorithms and experiments to obtain a better percentage of accuracy on heterogeneous data. Nevertheless, these solutions also suffer from non-iid data.

In addition, extensive experiments were performed, using publicly available datasets, with the proposed method that mitigates non-iid data impact through FedAvg-BE algorithm. This proposed algorithm provides Federated Averaging with the border entropy evaluation by selecting quality data with lower entropy from devices with non-iid data. This results in good computation and mitigation of the non-iid data effect in a FL environment. The experiment results demonstrate that our proposal enables an execution time reduction up to 22% for MNIST dataset and 26% for CIFAR-10 dataset, in the best cases.

The main contributions made in this work regarding the answer to the research question and the established objectives are summarized as follows.:

- Evaluation of entropy influence on non-iid data in a Federated Learning environment.
- Elaboration of a methodology for evaluating and selecting data, without causing a major impact on the communication between the devices.
- Development of a solution model that allows the creation of Federated Learning applications that evaluate the quality of the data in the client.
- Creation of an information processing algorithm that extends the Federated Learning application model by mitigating the effects of non-iid data.

This work is organized as follows. Section II provides a little background. Section III covers related work and an overview of some federated learning models. Section IV shows in detail the proposed model. The methodology, main experiments, and results are shown in Section V, and finally, Section VI presents the final considerations and future work.

II. BACKGROUND

This Section provides three main backgrounds: Non-iid Data, Entropy, and Federated Learning. A Federated Learning approach allows devices to learn a shared ML model collaboratively while maintaining data privacy. Non-iid data is the assumption that the data is unstructured, with a high degree of randomness. Entropy is a measure of a disorder found in the data; that is, it measures the degree of randomness of a variable.

A. IID VERSUS NON-IID DATA

In statistics, a set of random variables is iid when the data have the same probability distribution and are independent. In other words, data is more homogeneous [11]. For example, if it flips a coin 100 times, we would get heads 53 times and tails 47 times. However, if we toss the same coin for the 101st time, the likelihood is the same at 50% for each other. This means the result will still be heads or tails even if it does not save information from previous results.

Having independent and identically distributed data is one of the common assumptions for machine learning, statistical procedures, and hypothesis testing. However, in a Federated Learning environment, the devices are typically IoT devices composed of heterogeneous sensors that can get data with some associated bias. Due to this, non-iid data is a reality in this environment [12].

On the other hand, non-iid data is the assumption that the data is unstructured or loosely structured with a high degree of randomness. Objects, values, attributes and other everyday aspects are essential non-iid [13]. Figure 1 illustrates the difference between iid and non-iid data in an example dataset.

B. ENTROPY

The concept of entropy, which is part of one of the ideas of information theory, will be used throughout this work, as it is used to characterize probability distributions or quantify the similarity between information probability distributions. In computing, entropy measures the degree of randomness of data [8]. Therefore, it can be said that the higher the entropy, the more the data is non-iid. Figure 2 shows a visual example

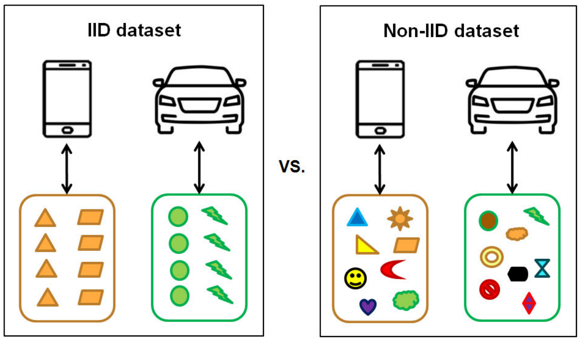


FIGURE 1. **Illustration of IID vs. Non-IID datasets.**

to represent entropy. Suppose there are three pots, one with red balls, one with yellow balls, and the third empty. The empty pot is filled with yellow balls; on top, the red balls are separated and organized by color. When shaking the jar, the balls mix so that there is no more separation, and they will hardly return to the initial state organized. That is, it became a system with a disorder, meaning an increase in entropy.

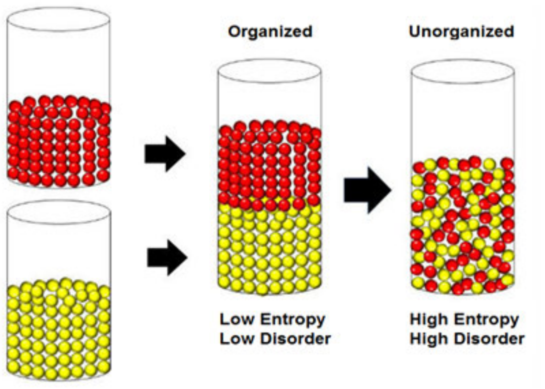


FIGURE 2. **Entropy example with marbles in jars.**

The chart in Figure 3 shows entropy for a database in which the classification is binary, with two possible values. It is also possible to calculate entropy for any number of values. The X-axis represents the probability of one of the classes occurring in the dataset. The Y axis is the result of the entropy of the dataset. The distribution is almost deterministic when p is closer to 0 because the random variable is almost always 0. The distribution is virtually deterministic when p is closer to 1 because the random variable is almost always 1. When $p = 0.5$, the entropy is maximum because the distribution is uniform over the results of the number of values.

The amount of uncertainty in an integer probability distribution can be quantified using the Shannon entropy formula. Therefore, the Shannon entropy of a distribution is the expected amount of information in an event extracted from that distribution, as illustrated in the equation below.

$$H = -\sum_{i=1}^{m} Pi.\log_2(Pi) \tag{1}$$

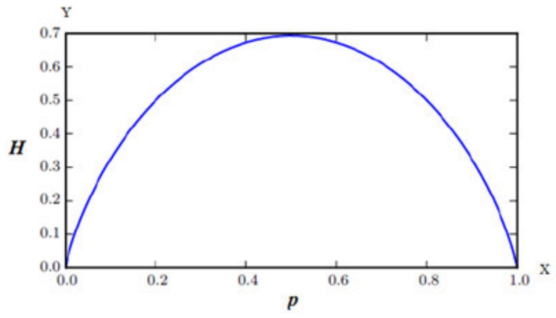


FIGURE 3. **Entropy chart for binary classification (adapted from [8].**

where:

H = Entropy, measurement of the degree of information uncertainty.

Pi = Probability that a fraction of records belong to class i in a dataset.

m = dataset size.

C. MACHINE LEARNING

In ML, types of Neural networks are widely studied due to their ability to analyze massive data sets. Convolutional Neural Network (CNN), for instance, is a chain where multiple convolution layers can make the network capable of recognizing complex constructs in large datasets [14]. This work chose to use CNN as a neural network model, as it is a type widely used in the literature for image recognition and processing pixel data.

Some algorithms are used to optimize the computation of neural networks, for example, Stochastic Gradient Descent (SGD), Adam, and RMSprop. SGD is a popular algorithm in various ML systems. SGD is a set of steps to achieve a minimum for a determined function [8]. Algorithms based on SGD minimize a defined loss function on model outputs, which adapt model parameters in the negative gradient direction. Gradient descent is called stochastic because the gradient is computed from an individual sample random subset of the training data [15]. Adaptive Moments (Adam) is an adaptive optimization algorithm for learning rate. The momentum parameter is incorporated directly as a first-order, exponentially weighted gradient estimate. This algorithm includes bias corrections for first- and second-order moment estimations for taking in to count at startup [16]. RMSprop is an algorithm that performs best in the non-convex configuration, changing the gradient buildup in an exponentially weighted moving average. In some algorithms, when applied to a non-convex function to train the neural network, learning can go through many different structures and arrive in a region where it is convex, like a bowl, so the learning rate may have become too small before reaching this region. The RMSProp algorithm uses an exponentially decaying average to discard extreme history so that it can quickly converge after finding a convex area, as it has a hyperparameter that controls the length of the moving average [8].

D. FEDERATED LEARNING

FL systems can be deployed where multiple parties jointly learn an accurate deep neural network, keeping the data itself local and confidential. There are scenarios where it is beneficial or even mandatory to isolate different subsets of training data from each other to maintain data privacy [15]. FL approaches allow for more intelligent models, low latency, lower power consumption, and guaranteed privacy. It works without storing user data in the Cloud Computing (CC).

In FL, each device trains its ML model locally in this approach. The update of parameters and weights is sent to the CC in encrypted form. In the CC, it is averaged with other user updates, thus improving the shared model [10]. Figure 4 illustrates how an FL model works. The data remains on the local device, and no sensitive data from the device is offloaded to the CC.

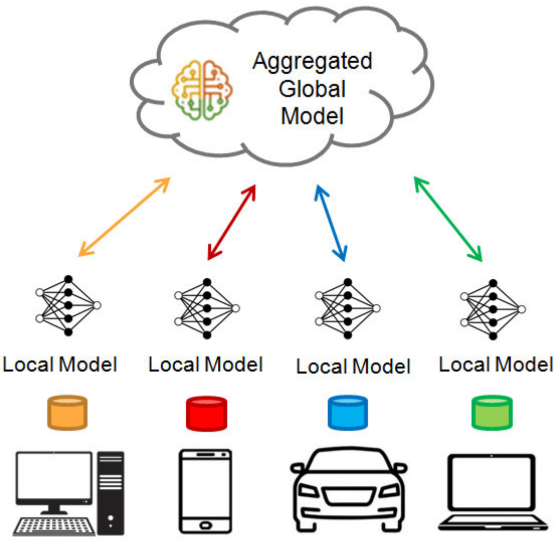


FIGURE 4. Federated Learning.

An example of FL application is the next-word prediction task on a cell phone. To perform this task, preserving the privacy of the text data and reducing the stress on the network, it is necessary to train a predictor in a distributed way instead of sending the raw data to a central server. In this configuration, remote devices periodically communicate with a central server to learn a global model. On each communication round, a subset of selected phones perform local training on their non-identically distributed user data and send these local updates to the server. After incorporating the updates, the server pushes the new global model back to another subset of devices. This iterative training process continues throughout the network until convergence is achieved [17].

The communication protocol comprises selection, configuration, reporting, training, and aggregation. However, this approach has data based on user usage. Thus, the datasets can suffer from non-iid, have an unbalanced behavior, be massively distributed, and present a limited communication channel. Therefore, decision-making may not reflect a global scenario, and the result may differ when selected devices perform many local update rounds.

Several examples of FL algorithms exist in the literature. The Federated Averaging algorithm (FedAvg) was proposed in 2016 by McMahan [18] based on data parallelism. FedAvg eliminates the need to upload sensitive user data to a centralized server, allowing edge devices to train a shared model locally on their local dataset. It aggregates local model updates (gradients), and FedAvg fulfills the essential privacy protection and data security requirements.

III. RELATED WORK

FL is a distributed machine learning approach that can achieve the purpose of collaborative learning from a large amount of data belonging to different parties and thus preserve the privacy of that data. The article [19] presents theoretical concepts and a brief survey of existing studies on FL and its use in data from wireless vehicular IoT sensors.

Studies have indicated that a deterioration in FL accuracy is almost inevitable with non-iid data. The performance degradation can be mainly attributed to the weight divergence of local models resulting from non-iid [20]. When divergence increases, it causes slower convergence, making learning worse.

In FL, various models have been proposed for collaborative learning of large amounts of data, which preserve data privacy. The authors in [18] present the FederatedAveraging (FedAvg) model. It is a practical method for Federated Learning of deep networks based on iterative averaging, which performs extensive empirical evaluation. They demonstrate in experiments that the approach is robust for unstructured and non-iid data distributions. Communication costs are the primary constraint and show a reduction in required communication rounds between 10 and 100 compared to synchronized SGD.

The work of Sahu [21] presents a framework, FedProx, to deal with heterogeneity in federated networks. FedProx can be seen as a generalization and reparameterization of FedAvg. However, this reparameterization makes only minor modifications to the method itself. The authors claim that FedProx allows for more robust convergence than FedAvg on a set of heterogeneous datasets, and FedProx demonstrates more stable and accurate convergence behavior.

Federated optimization models such as FedAvg are often difficult to tune and exhibit negative convergence behavior. The article by [22] proposes a model in the federated version of adaptive optimization, called FedOpt, where its convergence is analyzed in the presence of heterogeneous data for a non-convex configuration. The authors performed experiments and showed that adaptive optimizations improve accuracy in the Federated Learning environment.

When data is non-iid, the FedAvg model can drift on the client, resulting in instability and slow convergence. The article by Karimireddy [23] proposes a model with a stochastic controlled mean algorithm called SCAFFOLD, which uses control variables to reduce variance and correct for customer deviation in its local updates. This model requires fewer communication rounds and is not affected by data heterogeneity. It uses the similarity in customer data, resulting in faster convergence. According to the authors, SCAFFOLD outperforms FedAvg in non-convex experiments.

In data analysis, it is always assumed that the training data are iid. This assumption is sometimes incompatible with the real world because the data in users is often non-iid and may have a different probability distribution. Other recent research has addressed Federated Learning challenges with non-iid data. Recent works try to improve various aspects such as energy consumption of devices [36], data privacy [35] and entropy [37] to improve model effectiveness and accuracy based on basic FL structure and non-iid training data. Most works use CNN neural network model and use accuracy as a metric. Table 1 shows the list of related works and their characteristics.

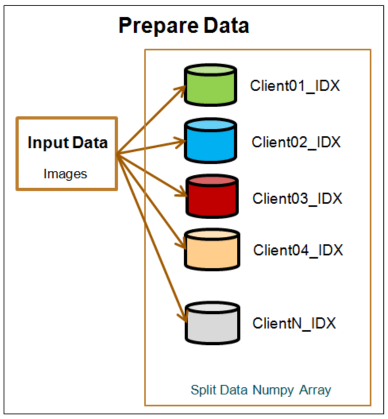
Among the works that try to improve data entropy, the authors of [25] present an FL method called FedEntropy with a dynamic device grouping scheme, which considers the distribution of heterogeneous devices and contributions of local models based on the judgment of maximum entropy. The work of [37] proposes an FL method with semi-supervised learning called DS-FL that improves the outputs with mean entropy reduction to prevent non-iid data from leading mobile devices to ambiguity and decreasing training convergence, this way the devices exchange these aggregated model outputs, instead of exchanging model parameters, to reduce communication cost. Both work do not propose filtering the non-iid data in the client with the lowest entropy concerning the average entropy of the labels, which is the proposal of this work.

The entropy measure can be directly related to non-iid data to optimize the SGD calculation and this work propose to use entropy computation as a data selection criterion to define training set and reduce model execution time.

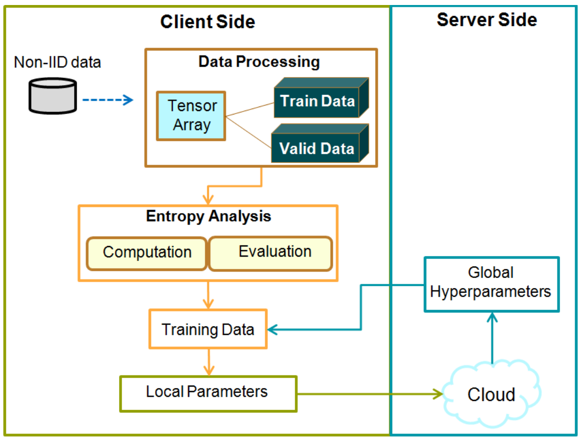
IV. MODEL

The IoT enables several critical services, such as self-driving cars in vehicular edge computing applications, where FL is a core for AI computation. The non-iid data influence, in this case, might contribute to slow make-decisions in critical situations conducting to live risk. Because of this, it is need providing algorithms and applications that mitigate such effects.

This work proposes a model that uses entropy computing as a data selection criterion in the local edge device by cross-device, defining a training set to avoid non-iid data. Thus, the amount of data is of higher quality due to lower entropy and can have lower uncertainty. The algorithm called FedAvg-BE, Federated Averaging Learning with the border entropy evaluation shows a methodology to evaluate border data using entropy to identify similar data with low aggregated information and select the best data into the dataset, which will provide relevant learning information for the computing of Federated Learning model.



(a) First Step



(b) Second Step

FIGURE 5. Proposed model.

Table 2 summarizes the notation used throughout this proposal method.

The method is composed of two main steps. The first step before running the training is data preparation. In this step, input data are images that will be transformed into numerical index matrices. These arrays will be randomly partitioned for each client into a file with an array of indexes called IDX, in Figure 5a. These indexes represent the images from the input data in non-iid format. From the partitioned data, the entropy is calculated for training data for each client, then this value is uploaded to the server, and the mean from all clients is computed to be saved as a global mean. After data partitioning, data processing on the client side performs, where each IDX index is transformed into a matrix of tensors. Then the tensor data will be split into training and validation data in 80% - 20% proportion. The proportion is an AI assumption and provides a consistent data analysis from the SGD computation model.

In the second step, in Figure 5b, the algorithm FedAvg-BE performs the entropy computation that will be compared with a global mean entropy and evaluated for each client before training execution. Data with lower entropy than the mean of

TABLE 1. Related works and their characteristics.

Work	Year	Author	Data Privacy	Energy Consumption	Entropy	IoT devices	Text Data	Image Data	Video Data	Neural Networks	Q-Learning	Clustering	Accuracy Metric	Connectivity Metric	Error Rate
FedAvg - Communication-Efficient Learning of Deep Networks from Decentralized Data	2017	[18]	X					X		X			X		
FedProx - Federated Optimization in Heterogeneous Networks	2020	[21]	X					X		X			X		
FedSwap - Semisupervised Distributed Learning With non-iid Data for AIoT Service Platform	2020	[13]	X			X		X		X			X		
Favor - Optimizing Federated Learning on non-iid Data with Reinforcement Learning	2020	[24]	X					X			X		X		
CSFedAvg - Client Selection for Federated Learning With non-iid Data in Mobile Edge Computing	2021	[12]	X					X		X			X		
FedOpt - ADAPTIVE FEDERATED OPTIMIZATION	2021	[22]						X		X			X		
SCAFFOLD - Stochastic Controlled Averaging for Federated Learning	2021	[23]						X		X			X		
FedEntropy - Efficient Device Grouping for Federated Learning Using Maximum Entropy Judgment	2022	[25]			X			X		X			X	X	
FedLog - Federated Anomaly Detection on System Logs for the Internet of Things: A Customizable and Communication-Efficient Approach	2022	[26]	X			X	X			X			X		
Adaptive Federated Learning on non-iid Data With Resource Constraint	2022	[27]						X		X			X		
AFPC - Adaptive Federated Deep Reinforcement Learning for Proactive Content Caching in Edge Computing	2022	[28]		X		X			X		X		X	X	
FEEL - Federated End-to-End Learning With non-iid Data for Vehicular Ad Hoc Networks	2022	[29]				X	X	X		X		X	X		X
GOF-TTE - Generative Online Federated Learning Framework for Travel Time Estimation	2022	[30]					X			X					X
A Novel Joint Dataset and Computation Management Scheme for Energy-Efficient Federated Learning in Mobile Edge Computing	2022	[31]		X				X		X			X		
Federated Learning With non-iid Data in Wireless Networks	2022	[32]		X				X		X			X	X	
FedHome - CC-Edge Based Personalized Federated Learning for In-Home Health Monitoring	2022	[33]	X			X	X			X			X		
FedMDS - An Efficient Model Discrepancy-Aware Semi-Asynchronous Clustered Federated Learning Framework	2023	[34]						X				X	X		
FedGS - Reschedule Gradients: Temporal non-iid Resilient Federated Learning	2023	[35]	X					X		X			X		
Energy-Efficient Multiprocessor-Based Computation and Communication Resource Allocation in Two-Tier Federated Learning Networks	2023	[36]		X		X		X		X			X		
DS-FL - Distillation-Based Semi-Supervised Federated Learning for Communication-Efficient Collaborative Training With non-iid Private Data	2023	[37]			X			X		X			X	X	

the global entropy will be selected and trained locally. On the other hand, it is discarded and not processed. After local training, accuracy is calculated from validation data. The local parameters will be obtained in the CC and sent to the server. On the server side, with the parameters received from each client, the global hyperparameters, including global accuracy, will be calculated and sent to the clients to retrain the data and obtain the local parameters. This cycle will be repeated for pre-defined epochs and rounds, and at the end of these, the result of the model's global accuracy calculation will be obtained.

Algorithm 1 presents the detailed steps to be implemented in the proposed model. First, the admin server initializes the global weight. For each round, the number of participating clients is determined. The client's data are selected based on the entropy computation, represented by equation 1.

TABLE 2. Notation adopted for the model description.

Symbol	Description
w	Weight model parameters
K	Client set
k	Client index
t	Number of round
C	Client Random fraction set
c	Fraction of client set
m	Number of selected clients
S	Random set of m clients
n	Data set size
β	Local batch set
P	Partitioned data
B	Local minibatch size
b	Coefficient from batch set
E	Number of local epochs
η	Learning rate
d	Data set for entropy computing
H	Entropy
p	Fraction probability

TABLE 3. Datasets adopted for the experiments.

Dataset	Description	Format	Scale	Training Set	Test Set
MNIST	Images of handwritten digits	28x28	Gray	60.000	10.000
CIFAR-10	Images from Canadian Institute For Advanced Research	32x32	RGB	50.000	10.000

Each selected client batch performs local training and returns the weights optimized by the ClientUpdate(k, w) training function. This function creates batches or partitions of customer data with many local epochs. For each epoch and partition, it trains the model locally. After receiving the parameters of each client, the global model is updated in the administrator. The proposed model in Figure 5 does not alter the computation of the Federated Learning model but adds a step in the pre-processing data on the client side, avoiding non-iid computing by the client. This strategy minimizes the non-iid influence of the Federated Learning model without providing substantial complexity to the current model.

V. RESULTS AND EVALUATIONS

A total of 115 experiments were performed to validate the hypothesis that lower entropy implies better data quality to mitigate non-iid data impact in a federated learning environment, using two publicly available datasets, MNIST and CIFAR-10, that consists of images in 10 classes.

Table 3 summarizes the datasets used in the experiments throughout this work.

For the experiment setup, the following items were used to generate the results of the experiments and thus allowing comparisons between the FL models:

Algorithm 1 FedAvg-BE. The K Clients Are Indexed by k; B Is the Local Minibatch Size, E Is the Number of Local Epochs, and η Is the Learning Rate

```
Require:
 1: initialize $w_0$
 2: $meanEntropy \leftarrow$ GetMeanEntropy($K$)
 3: for each round t = 1,2,... do
 4:     $m \leftarrow max(C.K, 1)$    ▷ Select the participated clients data
 5:     $S_t \leftarrow m$ (set of $m$ clients)
 6:     for $k \in S_t$ do
 7:         $w^k_{t+1} \leftarrow ClientUpdate(k, w_t)$    ▷ Update client model
 8:     $w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w^k_{t+1}$    ▷ Update the global model
 9: function ClientUpdate($k, w$)    ▷ Local computations
10:     $\beta \leftarrow$ (split $P_k$ into batches of size $B$)
11:     for each local epoch $i$ from 1 to $E$ do
12:         for batch $b \in \beta$ do
13:             $Hbatch \leftarrow$ Entropy($b$)    ▷ Entropy computation
14:             if $Hbatch < meanEntropy$ then
15:                 $w \leftarrow w - \eta\nabla\ell(w; b)$    ▷ Model training
16:     return $w$ to server
17: function GetMeanEntropy($k$)    ▷ Global entropy
18:     $meanEntropy \leftarrow 0.0$
19:     $totalEntropy \leftarrow 0.0$
20:     for each client $c \in k$ do
21:         $Hclient \leftarrow$ Entropy($c$)    ▷ Local entropy computation
22:         $totalEntropy \leftarrow totalEntropy + Hclient$
23:     $meanEntropy \leftarrow \frac{totalEntropy}{k}$    ▷ Global entropy computation
24:     return $meanEntropy$
25: function Entropy($d$)    ▷ Entropy dataset computation
26:     $H \leftarrow -\sum_d p(d) \log_2 p(d)$
27:     return $H$
```

- Centralized model, where execution is carried out on only one client, that is, not distributed, for comparison purposes with the following models;
- Four Federated Learning models: FedAvg [18], FedProx [21], FedOpt [22], Scaffold [23];
- The proposed FL model with entropy computing: FedAvg-BE;
- Software: Python 3.8 [38]
- Software to generate graphics: TensorBoard 2.10.0 [39]
- Framework: PyTorch 1.13.1 [40] – Framework that allows running with GPU and CPU, using the function called "cuda";
- Software Development Kit (SDK): NVFlare 2.0 [41] – NVIDIA Federated Learning Application Runtime Environment is an extensible, open source, domain-independent SDK for Federated Learning. This SDK

uses the Job concept, which allows you to run several experiments in an organized and agile way. In each Job, the application is deployed, in which the experiment will be executed;

- Neural Network: CNN – The choice of this type of neural network was because it is ordinarily chosen for images recognition and processing pixel data. Therefore, each dataset used in this work will have a customized CNN model. Figure 6 represent the neural network model to be implemented for the datasets. MNIST is a simpler dataset in GrayScale, so 3 convolutional layers were used. For CIFAR-10, 3 blocks of layers were needed and each block has 1 convolutional layer and 1 residual layer;
- Optimization Algorithms: SGD, Adam and RMSprop;
- Cloud: Openstack.
- Versioning: Github – to version and make publicly available the source codes in python of the Federated Learning applications, used for the study and experiments of this work.

The steps to implement the environment architecture for Federated Learning used in this work will be as follows.

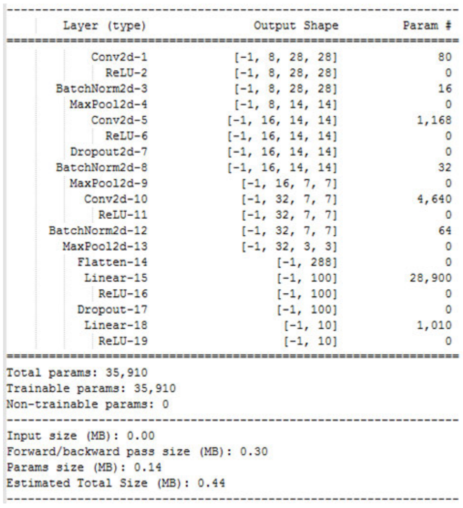
Virtual Machines (VMs) were created in the OpenStack tool, one server and nineteen clients on the same network. Each VM had 4 CPUs, Linux Ubuntu 18.04.4 LTS operating system, and Python 3.8 software installation. The VM configuration was as follows. The server had 8GB RAM and 100 GB storage. All 19 clients had 4GB RAM and 100 GB storage. The network engine was defined with NVFlare SDK and PyTorch framework.

Applications were developed for each Federated Learning model, including the FedAvg model adapted with entropy calculation. The Python classes were implemented in each application in NVFlare's network engine. For example, a Python file contains the CNN neural network model functions, and another includes the learner class, where variables are initialized, data is normalized and split into training and validation data. The code implementation files are available on https://github.com/fernanda-orlandi/dm-nvflare-fedavg-entropy.

A centralized service was established to manage training sessions. The application's config_fed_server.json file serves to configure server parameters to run the application with NVFlare. In this server file, the follow variables are defined: number of rounds, number of clients and a class of the Neural Network model. A client system was established to coordinate model parameters with the central service and share the best weights for entropy calculation. The application's config_fed_client.json file serves to configure the client's parameters to run the application with NVFlare. In this client file, the follow variables are defined: number of epochs, learning rate and a class of the Learner Executor.

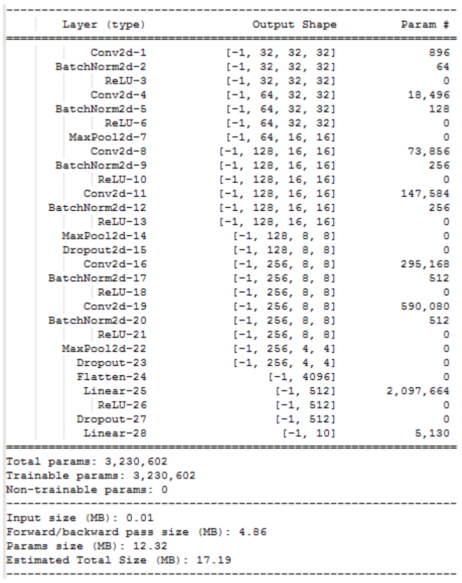
A. RESULTS

The 115 experiments were performed with the datasets and models mentioned in the previous section on



```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1            [-1, 8, 28, 28]              80
              ReLU-2            [-1, 8, 28, 28]               0
       BatchNorm2d-3            [-1, 8, 28, 28]              16
         MaxPool2d-4            [-1, 8, 14, 14]               0
            Conv2d-5           [-1, 16, 14, 14]           1,168
              ReLU-6           [-1, 16, 14, 14]               0
         Dropout2d-7           [-1, 16, 14, 14]               0
       BatchNorm2d-8           [-1, 16, 14, 14]              32
         MaxPool2d-9             [-1, 16, 7, 7]               0
           Conv2d-10             [-1, 32, 7, 7]           4,640
             ReLU-11             [-1, 32, 7, 7]               0
      BatchNorm2d-12             [-1, 32, 7, 7]              64
        MaxPool2d-13             [-1, 32, 3, 3]               0
          Flatten-14                  [-1, 288]               0
           Linear-15                  [-1, 100]          28,900
             ReLU-16                  [-1, 100]               0
          Dropout-17                  [-1, 100]               0
           Linear-18                   [-1, 10]           1,010
             ReLU-19                   [-1, 10]               0
================================================================
Total params: 35,910
Trainable params: 35,910
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.00
Forward/backward pass size (MB): 0.30
Params size (MB): 0.14
Estimated Total Size (MB): 0.44
----------------------------------------------------------------
```

(a) CNN for MNIST



```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1           [-1, 32, 32, 32]             896
       BatchNorm2d-2           [-1, 32, 32, 32]              64
              ReLU-3           [-1, 32, 32, 32]               0
            Conv2d-4           [-1, 64, 32, 32]          18,496
       BatchNorm2d-5           [-1, 64, 32, 32]             128
              ReLU-6           [-1, 64, 32, 32]               0
         MaxPool2d-7           [-1, 64, 16, 16]               0
            Conv2d-8          [-1, 128, 16, 16]          73,856
       BatchNorm2d-9          [-1, 128, 16, 16]             256
             ReLU-10          [-1, 128, 16, 16]               0
           Conv2d-11          [-1, 128, 16, 16]         147,584
      BatchNorm2d-12          [-1, 128, 16, 16]             256
             ReLU-13          [-1, 128, 16, 16]               0
        MaxPool2d-14            [-1, 128, 8, 8]               0
        Dropout2d-15            [-1, 128, 8, 8]               0
           Conv2d-16            [-1, 256, 8, 8]         295,168
      BatchNorm2d-17            [-1, 256, 8, 8]             512
             ReLU-18            [-1, 256, 8, 8]               0
           Conv2d-19            [-1, 256, 8, 8]         590,080
      BatchNorm2d-20            [-1, 256, 8, 8]             512
             ReLU-21            [-1, 256, 8, 8]               0
        MaxPool2d-22            [-1, 256, 4, 4]               0
          Dropout-23            [-1, 256, 4, 4]               0
          Flatten-24                 [-1, 4096]               0
           Linear-25                  [-1, 512]       2,097,664
             ReLU-26                  [-1, 512]               0
          Dropout-27                  [-1, 512]               0
           Linear-28                   [-1, 10]           5,130
================================================================
Total params: 3,230,602
Trainable params: 3,230,602
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.01
Forward/backward pass size (MB): 4.86
Params size (MB): 12.32
Estimated Total Size (MB): 17.19
----------------------------------------------------------------
```

(b) CNN for CIFAR-10

FIGURE 6. CNN summary for adopted datasets.

experiment setup. For the MNIST dataset, the accuracy average experiments were 98%, and for CIFAR-10, the average was 80%.

The strategy of doing the first experiments with the MNIST dataset was adopted, because the images are simpler to process in the neural network to make adjustments and stabilize the model algorithm with entropy calculation. Furthermore, the configuration consisted of 10 clients, 10 epochs, 50 rounds, SGD optimization algorithm, learning rate 0.01, CNN neural network class adjustments, and preparation of non-iid data for each client. In some experiments, the momentum value was modified between 0.9 and 0.98. First, the experiments were performed with the FedAvg model without entropy. Then, the model was changed to calculate the average entropy across all clients and select the data, with entropy less than the average, within the training method of each client. Hence, all clients continued to run in the federated environment and shared their parameters until the end of execution. Table 4 shows the accuracy and total time of these experiments.

TABLE 4. First experiments with adjustments to the CNN model and entropy computing for non-iid data.

Job	Dataset	Application	Accuracy (%)	Total Time (min)
20	MNIST	mnist_fedavg	98.83	65
21	MNIST	mnist_fedentropy	98.83	60
22	MNIST	mnist_fedentropy	98.61	60
23	MNIST	mnist_fedentropy	99.13	60
27	MNIST	mnist_fedavg	98.77	49
28	MNIST	mnist_fedavg	98.40	53
29	MNIST	mnist_fedavg	98.92	52
30	MNIST	mnist_fedentropy	98.81	53
31	MNIST	mnist_fedentropy	98.02	41
32	CIFAR10	cifar10_fedentropy	73.58	383

With the adjustments in the neural network model and entropy calculation, new experiments were carried out to compare some optimization algorithms with SGD. The algorithms chosen for the comparative tests were Adam and RMSprop. For these two algorithms, the Learning Rate parameter needed to be modified to 0.001, while in the SGD, it was set to 0.01 for the MNIST dataset. The experiments were also conducted to compare the FedAvg model results with the proposed FedAvg-BE model.

For the CIFAR-10 dataset, several experiments were also carried out with the FedAvg model with and without entropy, running with the variation of the optimization algorithms SGD, Adam, and RMSprop. The adjustment for the neural network model was more complex for this dataset, so several tests were necessary to obtain an adequate neural network. Image processing of the CIFAR-10 dataset requires a neural network model with three blocks. Each block has two convolutional layers, one of which is a residual layer, which requires a longer processing time, but provides greater accuracy. There was better convergence when the Leaning Rate parameter was changed to 0.001 in the execution with the SGD algorithm and to 0.0001 in the execution with the Adam and RMSprop algorithms. For Adam and RMSprop, momentum parameter was not configured. For both models, with and without entropy calculation, the SGD algorithm showed better accuracy results for the CIFAR10 dataset.

Table 5 presents the results of the most relevant experiments for 10 clients with the MNIST and CIFAR-10 datasets, which compares the optimization algorithms and some parameter variations for the datasets. Compared to the SDG, Adam, and RMSprop optimization algorithms, it is noted that the SGD algorithm obtained better accuracy, both for the MNIST dataset and for the CIFAR-10 dataset, considering the non-iid data in each client.

For both datasets, it was observed that there was performance degradation when changing the following parameters: Learning Rate, Momentum, number of epochs.

TABLE 5. Most relevant experiments for MNIST and CIFAR-10 datasets, using non-iid data in 10 clients.

Job	Dataset	Model	Optimiz. Algo.	Acc. (%)	Total Time (min)
28	MNIST	FedAvg	SGD	**98.40**	**53**
31	MNIST	FedAvg-BE	SGD	**98.02**	**41**
41	MNIST	FedAvg	Adam	97.73	55
33	MNIST	FedAvg-BE	Adam	94.76	41
39	MNIST	FedAvg	RMSprop	97.41	60
37	MNIST	FedAvg-BE	RMSprop	93.27	42
51	CIFAR10	FedAvg	SGD	**85.36**	**540**
54	CIFAR10	FedAvg-BE	SGD	**80.74**	**400**
79	CIFAR10	FedAvg	Adam	74.17	540
82	CIFAR10	FedAvg-BE	Adam	58.88	339
78	CIFAR10	FedAvg	RMSprop	60.18	526
83	CIFAR10	FedAvg-BE	RMSprop	53.22	396

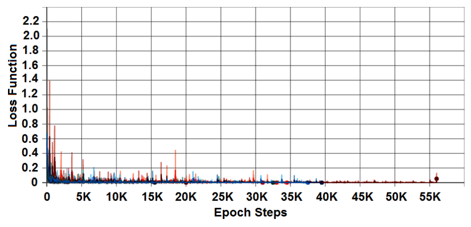
The Figures 7 and 8 represent comparative results of experiments with FedAvg model and proposed FedAvg-BE model for the MNIST dataset. The Figures 9 and 10 represent comparative results for the CIFAR-10 dataset. These experiments were performed with the configuration of 10 epochs and 50 rounds, in 10 clients. The non-iid data are heterogeneous and responsible for the high oscillation in the loss function results.

In both Figures, in the Global Accuracy and Loss Function lines, it is possible to visualize the graphs and compare the result of the FedAvg model with the same model adapted with entropy calculation, FedAvg-BE. When the entropy calculation is applied, it is possible to observe in the graphs that oscillation occurs in the convergence of the optimization algorithm and the loss function. This was expected, as fewer data is processed in training, regardless of the dataset type.

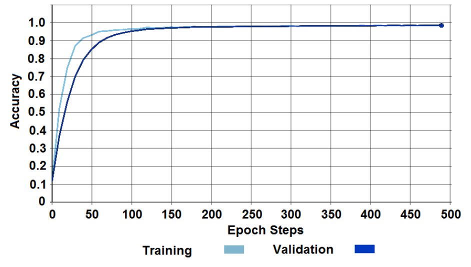
Note results for Jobs 28 and 31 in Table 5 for the MNIST dataset, that there was a reduction in execution time of 12 minutes, which represents a reduction of around 22%, and a reduction of 0.38% for accuracy. For the CIFAR-10 dataset, in Table 5 is possible to see the results for Jobs 51 and 54, where there was a reduction in execution time of around 2 hours, which represents a reduction of around 26%, and a reduction of 4.62% for accuracy.

The graphs 8 and 10 with the Global Accuracy and Loss Function showed a slight difference in the convergence of the SGD optimization algorithm and in the loss function due to data selection when entropy calculation was applied. With this, it is possible to compare the results of the experiments with the FedAvg model and with the model adapted with entropy calculation, FedAvg-BE, to select data with lower entropy.

There was an increase in the number of clients to 19, maintaining 10 epochs and 50 rounds, adjustments in the CNN neural network model class, and new preparation of Non-IID data for each client. The Learning Rate parameter was set to 0.01 for SGD, while for Adam and RMSprop, it was set to 0.001 for both datasets. After these changes, tests were performed with the FedAvg model without entropy changes and then with entropy calculation for each

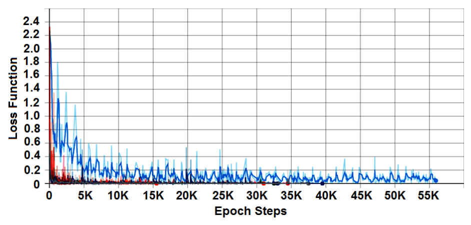


(a) Job 28 - Loss Function - FedAvg

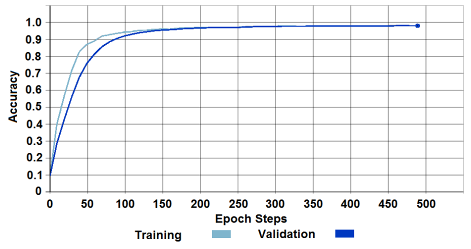


(b) Job 28 - Global Accuracy - FedAvg

FIGURE 7. Results for MNIST non-iid with SGD in 10 clients - Job 28.

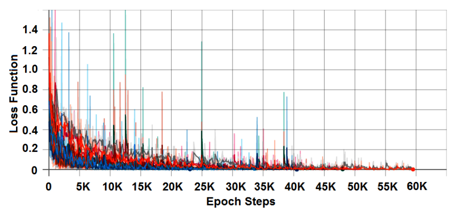


(a) Job 31 - Loss Function - FedAvg-BE

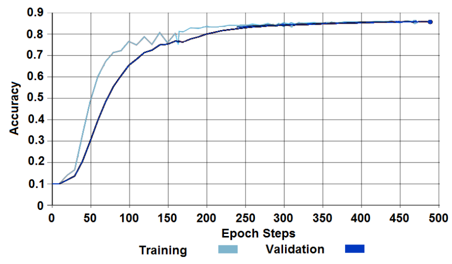


(b) Job 31 - Global Accuracy - FedAvg-BE

FIGURE 8. Results for MNIST non-iid with SGD in 10 clients - Job 31.

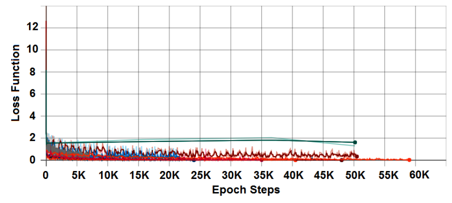


(a) Job 51 - Loss Function - FedAvg

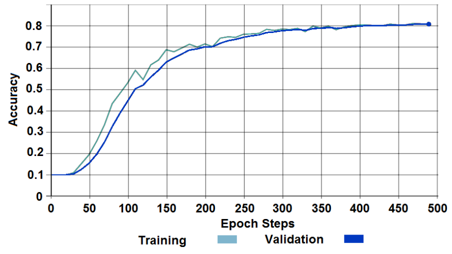


(b) Job 51 - Global Accuracy - FedAvg

FIGURE 9. Results for CIFAR-10 non-iid with SGD in 10 clients - Job 51.



(a) Job 54 - Loss Function - FedAvg-BE



(b) Job 54 - Global Accuracy - FedAvg-BE

FIGURE 10. Results for CIFAR-10 non-iid with SGD in 10 clients - Job 54.

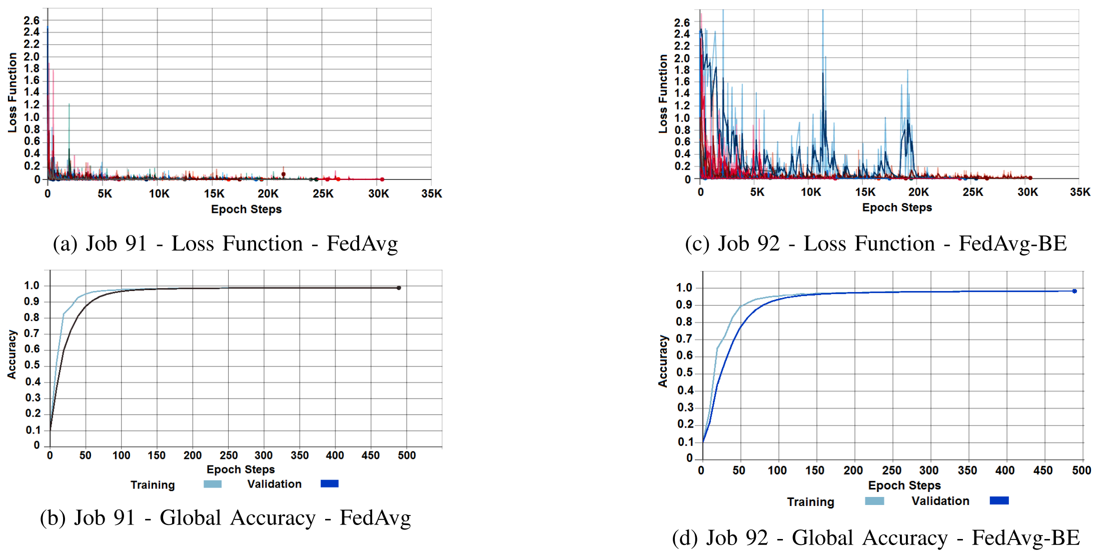
TABLE 6. Most relevant experiments for MNIST and CIFAR-10 datasets, using non-iid data in 19 clients.

Job	Dataset	Model	Optimiz. Algo.	Acc. (%)	Total Time (min)
91	MNIST	FedAvg	SGD	**98.88**	**39**
92	MNIST	FedAvg-BE	SGD	**98.31**	**34**
97	MNIST	FedAvg	Adam	98.43	41
98	MNIST	FedAvg-BE	Adam	98.08	35
99	MNIST	FedAvg	RMSprop	98.19	42
100	MNIST	FedAvg-BE	RMSprop	93.01	36
106	CIFAR10	FedAvg	SGD	**83.47**	**380**
107	CIFAR10	FedAvg-BE	SGD	**78.68**	**350**
108	CIFAR10	FedAvg	Adam	77.81	393
109	CIFAR10	FedAvg-BE	Adam	71.99	375
110	CIFAR10	FedAvg	RMSprop	70.21	424
111	CIFAR10	FedAvg-BE	RMSprop	60.68	386

optimization algorithm. Again, with and without entropy calculation, the SGD algorithm showed better accuracy results for the CIFAR-10 dataset. Table 6 lists the sequence of experiments performed with 19 clients for the MNIST and CIFAR-10 datasets.

Considering the 19 clients, Figures 11 and 12 show comparisons of the results of the experiments with the FedAvg model and the proposed FedAvg-BE model. Figure 11 represents results for the MNIST dataset, and Figure 12 for the CIFAR-10 dataset. These experiments were performed maintaining the configuration of 10 epochs and 50 rounds. The non-iid data are heterogeneous and responsible for the oscillation in the loss function results.

In both Figures, in the Global Accuracy and Loss Function lines, it is possible to visualize the graphs and compare the result of the FedAvg model with the FedAvg-BE model.



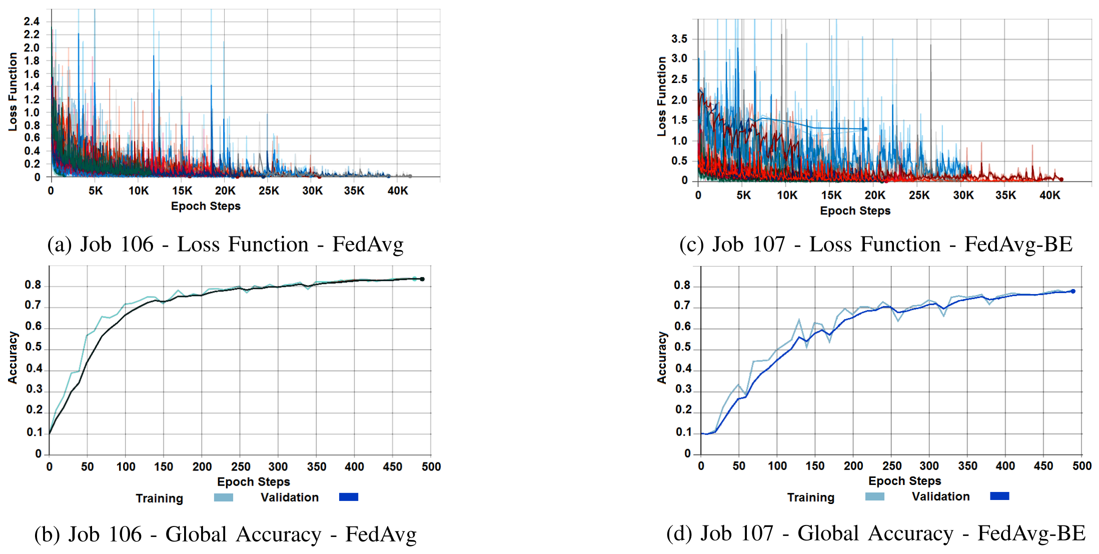
(a) Job 91 - Loss Function - FedAvg

(b) Job 91 - Global Accuracy - FedAvg

(c) Job 92 - Loss Function - FedAvg-BE

(d) Job 92 - Global Accuracy - FedAvg-BE

FIGURE 11. Results for MNIST non-iid with SGD in 19 clients.



(a) Job 106 - Loss Function - FedAvg

(b) Job 106 - Global Accuracy - FedAvg

(c) Job 107 - Loss Function - FedAvg-BE

(d) Job 107 - Global Accuracy - FedAvg-BE

FIGURE 12. Results for CIFAR-10 non-iid with SGD in 19 clients.

There was oscillation in the convergence of the optimization algorithm and the loss function. Increasing the number of clients to 19 clients did not affect the results, which are similar to those obtained with ten clients.

For the 19 clients, Table 6 and Figures 11 and 12 show the variation of the results obtained with the SGD algorithm, which showed a better result in accuracy, for the execution of the FedAvg model and for the FedAvg-BE model, which includes entropy calculation. Note that for the MNIST dataset, there was a reduction in execution time of 5 minutes, representing a reduction of around 12% and a reduction of 0.57% for accuracy. For the CIFAR-10 dataset, there was a reduction in execution time of approximately 30 minutes, representing a reduction of around 8% and a reduction of 4.79% for accuracy. Again, the graphs shows a comparative of the experiments results with the FedAvg model and FedAvg-BE model.

With the increase in the number of clients and the preparation of non-iid data for them, there was a more significant heterogeneity of the data distributed among the clients, which did not affect the accuracy result, with only a slight difference in the execution time between 10 and 19 clients, maintaining the reduction of it. The runtime reduction helps to lower energy consumption in an IoT device, for example. With that, the results with the FedAvg model adapted with entropy calculation, FedAvg-BE, proved acceptable and consistent.

VI. CONCLUSION AND FUTURE WORK

This work presents an experimental evaluation of Federated Learning models to support the hypothesis that entropy is directly related to non-iid data and compares Federated Learning models, implementing entropy calculation, to mitigate the impact of non-iid data in training. With the implementation of the FedAvg model adapted with entropy calculation, using the FedAvg-BE algorithm, experiments were performed with a variation of a dataset, CNN neural network, optimization algorithms, entropy calculation, and the number of clients.

Most model training performed with the SGD optimization algorithm tends to be more accurate than the Adam and RMSprop algorithms for both MNIST and CIFAR10 datasets. When using entropy calculation, fewer data are selected and processed, slightly reducing the accuracy between 1% and 5% and the execution time of the model between 12% and 26%, regardless of the dataset.

Finally, it is safe to say that the FedAvg-BE model, compared with the other FL models studied in this work, presents itself as the best choice to reduce the execution time of Federated Learning applications and helps to lower energy consumption for IoT devices. The results of the experiments were acceptable and consistent with the study proposal of this work.

Although the presented results have fulfilled the previously established objectives, future work is needed to optimize the model's accuracy and apply hardware with more resources making using a dataset with larger images possible. Furthermore includes tuning the FedAvg-BE model to maximize accuracy, perhaps combining another Federated Learning model. It is also intended to perform experiments with new available or customized datasets. Again, FedAvg-BE can be applied in a configuration with processing support for GPU-distributed training and with some tools to monitor power consumption for future experiments.

ACKNOWLEDGMENT

The experiments in this paper were carried out and conducted with the support of CEI, which belongs to the Institute of Informatics of UFRGS.

REFERENCES

[1] T. Shao, X. Kui, P. Zhang, and H. Chen, "Collaborative learning for answer selection in question answering," *IEEE Access*, vol. 7, pp. 7337–7347, 2019, doi: 10.1109/ACCESS.2018.2890102.

[2] O. Simeone, "A very brief introduction to machine learning with applications to communication systems," *IEEE Trans. Cogn. Commun. Netw.*, vol. 4, no. 4, pp. 648–664, Dec. 2018, doi: 10.1109/TCCN.2018.2881442.

[3] Z. Qu, Z. Zhang, B. Liu, P. Tiwari, X. Ning, and K. Muhammad, "Quantum detectable Byzantine agreement for distributed data trust management in blockchain," *Inf. Sci.*, vol. 637, Aug. 2023, Art. no. 118909, doi: 10.1016/j.ins.2023.03.134.

[4] J. C. S. D. Anjos, K. J. Matteussi, P. R. R. De Souza, G. J. A. Grabher, G. A. Borges, J. L. V. Barbosa, G. V. González, V. R. Q. Leithardt, and C. F. R. Geyer, "Data processing model to perform big data analytics in hybrid infrastructures," *IEEE Access*, vol. 8, pp. 170281–170294, 2020, doi: 10.1109/ACCESS.2020.3023344.

[5] Z. Xiao, H. Fang, H. Jiang, J. Bai, V. Havyarimana, H. Chen, and L. Jiao, "Understanding private car aggregation effect via spatio-temporal analysis of trajectory data," *IEEE Trans. Cybern.*, vol. 53, no. 4, pp. 2346–2357, Apr. 2023, doi: 10.1109/TCYB.2021.3117705.

[6] H. Hellström, J. Mairton B. da Silva Jr., M. Mohammadi Amiri, M. Chen, V. Fodor, H. V. Poor, and C. Fischione, "Wireless for machine learning," 2020, *arXiv:2008.13492*.

[7] Z. Shao, Q. Zhai, and X. Guan, "Physical-model-aided data-driven linear power flow model: An approach to address missing training data," *IEEE Trans. Power Syst.*, vol. 38, no. 3, pp. 2970–2973, May 2023, doi: 10.1109/TPWRS.2023.3256120.

[8] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning* (Adaptative Computation and Machine Learning Series), vol. 1, 1st ed. Cambridge, MA, USA: MIT Press, 2017.

[9] M. Li, Z. Tian, X. Du, X. Yuan, C. Shan, and M. Guizani, "Power normalized cepstral robust features of deep neural networks in a cloud computing data privacy protection scheme," *Neurocomputing*, vol. 518, pp. 165–173, Jan. 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231222013741, doi: 10.1016/j.neucom.2022.11.001.

[10] B. McMahan and R. S. D. Ramage, "Federated learning: Collaborative machine learning without centralized training data," Blog Online Google Res., Tech. Rep., Jul. 2023. [Online]. Available: https://ai.googleblog.com/2017/04/federated-learning-collaborative.html

[11] A. Clauset, "Inference, models and simulation for complex systems," Tech. Rep., 2011. Accessed: Mar. 2023. [Online]. Available: https://aaronclauset.github.io/courses/7000/csci7000-001_2011_L0.pdf

[12] W. Zhang, X. Wang, P. Zhou, W. Wu, and X. Zhang, "Client selection for federated learning with non-IID data in mobile edge computing," *IEEE Access*, vol. 9, pp. 24462–24474, 2021, doi: 10.1109/ACCESS.2021.3056919.

[13] T.-C. Chiu, Y.-Y. Shih, A.-C. Pang, C.-S. Wang, W. Weng, and C.-T. Chou, "Semisupervised distributed learning with non-IID data for AIoT service platform," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 9266–9277, Oct. 2020, doi: 10.1109/JIOT.2020.2995162.

[14] L. Alzubaidi, J. Zhang, A. J. Humaidi, A. Al-Dujaili, Y. Duan, O. Al-Shamma, J. Santamaría, M. A. Fadhel, M. Al-Amidie, and L. Farhan, "Review of deep learning: Concepts, CNN architectures, challenges, applications, future directions," *J. Big Data*, vol. 8, no. 1, pp. 1–74, Mar. 2021, doi: 10.1186/s40537-021-00444-8.

[15] J. Verbraeken, M. Wolting, J. Katzy, J. Kloppenburg, T. Verbelen, and J. S. Rellermeyer, "A survey on distributed machine learning," *ACM Comput. Surv.*, vol. 53, no. 2, pp. 1–33, Mar. 2021, doi: 10.1145/3377454.

[16] J. Jiang, B. Cui, and C. Zhang, *Distributed Machine Learning and Gradient Optimization* (Big Data Management). Singapore: Springer, 2022, doi: 10.1007/978-981-16-3420-8.

[17] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Process. Mag.*, vol. 37, no. 3, pp. 50–60, May 2020, doi: 10.1109/MSP.2020.2975749.

[18] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. 20th Int. Conf. Artif. Intell. Statist.*, vol. 54, A. Singh and J. Zhu. Eds., Fort Lauderdale, FL, USA, 2017, pp. 1273–1282.

[19] Z. Du, C. Wu, T. Yoshinaga, K. A. Yau, Y. Ji, and J. Li, "Federated learning for vehicular Internet of Things: Recent advances and open issues," *IEEE Open J. Comput. Soc.*, vol. 1, pp. 45–61, 2020, doi: 10.1109/OJCS.2020.2992630.

[20] H. Zhu, J. Xu, S. Liu, and Y. Jin, "Federated learning on non-IID data: A survey," *Neurocomputing*, vol. 465, pp. 371–390, Nov. 2021, doi: 10.1016/j.neucom.2021.07.098.

[21] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," in *Proc. Mach. Learn. Syst. (MLSys)*, vol. 2, I. S. Dhillon, D. S. Papailiopoulos, and V. Sze, Eds., Austin, TX, USA, 2020, pp. 429–450.

[22] S. Reddi, Z. Charles, M. Zaheer, Z. Garrett, K. Rush, J. Konečný, S. Kumar, and H. B. McMahan, "Adaptive federated optimization," 2020, *arXiv:2003.00295*.

[23] S. P. Karimireddy, S. Kale, M. Mohri, S. J. Reddi, S. U. Stich, and A. T. Suresh, "SCAFFOLD: Stochastic controlled averaging for federated learning," 2019, *arXiv:1910.06378*.

[24] H. Wang, Z. Kaplan, D. Niu, and B. Li, "Optimizing federated learning on non-IID data with reinforcement learning," in *Proc. IEEE Conf. Comput. Commun. (IEEE INFOCOM)*, Jul. 2020, pp. 1698–1707, doi: 10.1109/INFOCOM41043.2020.9155494.

[25] Z. Ling, Z. Yue, J. Xia, M. Hu, T. Wang, and M. Chen, "FedEntropy: Efficient device grouping for federated learning using maximum entropy judgment," 2022, *arXiv:2205.12038*.

[26] B. Li, S. Ma, R. Deng, K. R. Choo, and J. Yang, "Federated anomaly detection on system logs for the Internet of Things: A customizable and communication-efficient approach," *IEEE Trans. Netw. Serv. Manage.*, vol. 19, no. 2, pp. 1705–1716, Jun. 2022, doi: 10.1109/TNSM.2022.3152620.

[27] J. Zhang, S. Guo, Z. Qu, D. Zeng, Y. Zhan, Q. Liu, and R. Akerkar, "Adaptive federated learning on non-IID data with resource constraint," *IEEE Trans. Comput.*, vol. 71, no. 7, pp. 1655–1667, Jul. 2022, doi: 10.1109/TC.2021.3099723.

[28] D. Qiao, S. Guo, D. Liu, S. Long, P. Zhou, and Z. Li, "Adaptive federated deep reinforcement learning for proactive content caching in edge computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 12, pp. 4767–4782, Dec. 2022, doi: 10.1109/TPDS.2022.3201983.

[29] B. Li, Y. Jiang, Q. Pei, T. Li, L. Liu, and R. Lu, "FEEL: Federated end-to-end learning with non-IID data for vehicular ad hoc networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 9, pp. 16728–16740, Sep. 2022, doi: 10.1109/TITS.2022.3190294.

[30] Z. Zhang, H. Wang, Z. Fan, J. Chen, X. Song, and R. Shibasaki, "GOF-TTE: Generative online federated learning framework for travel time estimation," *IEEE Internet Things J.*, vol. 9, no. 23, pp. 24107–24121, Dec. 2022, doi: 10.1109/JIOT.2022.3190864.

[31] J. Kim, D. Kim, J. Lee, and J. Hwang, "A novel joint dataset and computation management scheme for energy-efficient federated learning in mobile edge computing," *IEEE Wireless Commun. Lett.*, vol. 11, no. 5, pp. 898–902, May 2022, doi: 10.1109/LWC.2022.3147236.

[32] Z. Zhao, C. Feng, W. Hong, J. Jiang, C. Jia, T. Q. S. Quek, and M. Peng, "Federated learning with non-IID data in wireless networks," *IEEE Trans. Wireless Commun.*, vol. 21, no. 3, pp. 1927–1942, Mar. 2022, doi: 10.1109/TWC.2021.3108197.

[33] Q. Wu, X. Chen, Z. Zhou, and J. Zhang, "FedHome: Cloud-edge based personalized federated learning for in-home health monitoring," *IEEE Trans. Mobile Comput.*, vol. 21, no. 8, pp. 2818–2832, Aug. 2022, doi: 10.1109/TMC.2020.3045266.

[34] Y. Zhang, D. Liu, M. Duan, L. Li, X. Chen, A. Ren, Y. Tan, and C. Wang, "FedMDS: An efficient model discrepancy-aware semi-asynchronous clustered federated learning framework," *IEEE Trans. Parallel Distrib. Syst.*, vol. 34, no. 3, pp. 1007–1019, Mar. 2023, doi: 10.1109/TPDS.2023.3237752.

[35] X. You, X. Liu, N. Jiang, J. Cai, and Z. Ying, "Reschedule gradients: Temporal non-IID resilient federated learning," *IEEE Internet Things J.*, vol. 10, no. 1, pp. 747–762, Jan. 2023, doi: 10.1109/JIOT.2022.3203233.

[36] R. Ruby, H. Yang, F. A. P. de Figueiredo, T. Huynh-The, and K. Wu, "Energy-efficient multiprocessor-based computation and communication resource allocation in two-tier federated learning networks," *IEEE Internet Things J.*, vol. 10, no. 7, pp. 5689–5703, Apr. 2023, doi: 10.1109/JIOT.2022.3153996.

[37] S. Itahara, T. Nishio, Y. Koda, M. Morikura, and K. Yamamoto, "Distillation-based semi-supervised federated learning for communication-efficient collaborative training with non-IID private data," *IEEE Trans. Mobile Comput.*, vol. 22, no. 1, pp. 191–205, Jan. 2023, doi: 10.1109/TMC.2021.3070013.

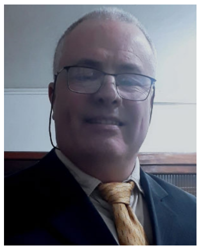
[38] Python. (Mar. 2023). *Python Programming Language*. [Online]. Available: https://docs.python.org/pt-br/3.8/

[39] TensorFlow. (Mar. 2023). *TensorBoard: TensorFlow's Visualization Toolkit*. [Online]. Available: https://www.tensorflow.org/tensorboard/

[40] PyTorch. (Mar. 2023). *PyTorch: Optimized Tensor Library for Deep Learning*. [Online]. Available: https://pytorch.org/docs/stable/index.html

[41] Nvidia. (Mar. 2023). *NVIDIA Federated Learning Application Runtime Environment*. [Online]. Available: https://developer.nvidia.com/flare

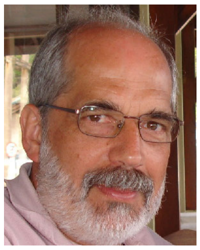
FERNANDA C. ORLANDI is currently pursuing the master's degree in computing with the Institute of Informatics, UFRGS/PPGC, Federal University of Rio Grande do Sul, Porto Alegre, Brazil. Her research interests include machine learning, big data, and federated learning.



JULIO C. S. DOS ANJOS received the bachelor's degree in electrical engineering from PUC/RS, in 1991, and the master's and Ph.D. degrees in computer science from UFRGS, in 2012 and 2017, respectively. He held a postdoctoral position in computer science with UFRGS, in 2021. Since May 2022, he has been a Professor with the Graduate Program in Teleinformatics Engineering, Federal University of Ceará (PPGETI/UFC), Brazil. His research interests include distributed systems, hybrid infrastructures, collaborative learning, intelligent autonomous systems, and big data analytics with deep learning.

VALDERI R. Q. LEITHARDT (Senior Member, IEEE) received the Ph.D. degree in informatics from INF-UFRGS, Brazil. He is currently an Associate Professor with the Polytechnic Institute of Portalegre and a Researcher with the VALORIZA Research Group, School of Technology and Management. He is a Research Collaborator with the Expert Systems and Applications Laboratory Group, University of Salamanca. His research interests include distributed systems, with a focus on data privacy, communication, and programming protocols, involving scenarios and applications for the Internet of Things, precision agriculture, smart cities, big data, cloud computing, and blockchain.

JUAN FRANCISCO DE PAZ SANTANA received the degree in technical engineering in systems computer sciences, in 2003, and the engineering degree in computer sciences, the degree in statistics, and the Ph.D. degree in computer science from the University of Salamanca, Spain, in 2005, 2007, and 2010, respectively. He is currently a Full Professor with the University of Salamanca, where he is also a Researcher with the Expert Systems and Applications Laboratory (ESALab). He is the coauthor of published articles in several journals, workshops, and symposiums.



CLAUDIO F. R. GEYER received the degree in mechanical engineering and the M.Sc. degree in computer science from UFRGS/RS, in 1978 and 1986, respectively, and the Ph.D. degree in informatics from Université de Grenoble I, in 1991. He is currently a Tenured Professor of computer science with UFRGS/RS. His research interests include pervasive and ubiquitous computing, grid and volunteer computing, scheduling and data-intensive computing, and big data.