Received 19 February 2024, accepted 27 March 2024, date of publication 1 April 2024, date of current version 5 April 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3383783

RESEARCH ARTICLE

Experimenting With Normalization Layers in Federated Learning on Non-IID Scenarios

BRUNO CASELLA[1], ROBERTO ESPOSITO[1], ANTONIO SCIARAPPA[2], CARLO CAVAZZONI[2], AND MARCO ALDINUCCI[1]
[1]Department of Computer Science, University of Turin, 10124 Turin, Italy
[2]Leonardo S.p.A., 00195 Rome, Italy

Corresponding author: Bruno Casella (bruno.casella@unito.it)

This work was supported in part by the Spoke ''FutureHPC & BigData'' of the ICSC–Centro Nazionale di Ricerca in ''High Performance Computing, Big Data and Quantum Computing,'' funded by European Union–NextGenerationEU; and in part by European Union within the H2020 RIA ''European Processor Initiative—Specific Grant Agreement 2'' (https://www.european-processor-initiative.eu/) under Grant 826647.

ABSTRACT Training Deep Learning (DL) models require large, high-quality datasets, often assembled with data from different institutions. Federated Learning (FL) has been emerging as a method for privacy-preserving pooling of datasets employing collaborative training from different institutions by iteratively globally aggregating locally trained models. One critical performance challenge of FL is operating on datasets not independently and identically distributed (non-IID) among the federation participants. Even though this fragility cannot be eliminated, it can be debunked by a suitable optimization of two hyper-parameters: layer normalization methods and collaboration frequency selection. In this work, we benchmark five different normalization layers for training Neural Networks (NNs), two families of non-IID data skew, and two datasets. Results show that Batch Normalization, widely employed for centralized DL, is not the best choice for FL, whereas Group and Layer Normalization consistently outperform Batch Normalization, with a performance gain of up to about 15 % in the most challenging non-IID scenario. Similarly, frequent model aggregation decreases convergence speed and mode quality.

INDEX TERMS Batch normalization, epochs per round, federated averaging, federated learning, neural networks, non-IID data, normalization layers.

I. INTRODUCTION

The constant development of Information and Communication Technologies has boosted the availability of computational resources and data, leading us to the Big Data era, where data-driven approaches have become a fundamental aspect of everyday decisions. Both computational resources and data are ubiquitous and inherently distributed. All public and private sectors, from scientific research to companies, take benefit from a vast amount of diverse data to support the growth of their business and to develop more accurate Artificial Intelligence (AI) systems.

Data is often spread and segregated in silos across different institutions and even different business units of the same organization. It is essential to make data accessible to all the partners to train high-quality models and exploit the entire data's value [1]. Many recent open science works have encouraged data sharing between institutions in order to improve research possibilities, create collaborations, and publish reproducible results. For example, data sharing across countries has been a crucial information tool during the COVID-19 pandemic [2].

However, data is often not shareable due to issues like privacy, security, ownership, trust, and economic reasons. For instance, the European regulation GDPR [3] places stringent constraints on the possibility of sharing sensitive data between parties; industrial companies do not share their data because leveraging it is seen as a competitive advantage. Also, exposing data to other institutions can raise concerns like lack of ownership and lack of trust.

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

To address these problems, model-sharing strategies (MSS) have emerged as an alternative to data sharing. In MSS, the idea is to share AI models between the involved parties in order to achieve collaboration without sharing raw data. In these approaches, the AI model can range from simpler Machine Learning (ML) algorithms like linear regression to more complex models such as those learned by Deep Learning techniques using Neural Networks (NNs). Recent years have seen the growth of different model-sharing approaches ranging from the "model-to-data remote access" approaches to Federated Learning [4]. In "model-to-data remote access" approaches, AI models are run remotely directly on the machines that hold the data, and security is enforced by leveraging secure remote connections and Trusted Execution Environments (TEEs) enclaves. Federated Learning has also emerged as a popular approach. In FL, the involved parties collaborate by aggregating locally trained models into a globally shared one. The process is usually iterative and based on NNs (FedAvg) [4], even if recently methods based on non-NN distributed boosting algorithms have been proposed [24]. These algorithms allow parties to aggregate any kind of model without making assumptions about the kind of model being aggregated or assuming a training procedure based on gradient descent [29].

FL is a distributed ML technique originally proposed by Google in 2016 to deal with sensitive data of mobile devices [4]. FL is an iterative version of model-sharing: clients (the data owners) create a federation (hence the name) together with the server and build a shared model based on the following steps: *1)* clients send their metadata, like the number of classes, training set size, test set size and shape of the input features, to the server, that initializes a model based on the received metadata characteristics; *2)* the server sends the initialized model to all the participants of the federation; *3)* after performing one or more steps of gradient descent, clients send the trained model back to the server; *4)* server acts as an aggregator performing a combination (a function like average, sum, maximum, minimum and so on) of the received models. The aggregated model is now sent to the clients, and steps *3)* and *4)* are repeated until a specified number of rounds are performed or a convergence criterion is met. The first proposed FL algorithm is the FedAvg [4] algorithm, where the aggregation function used to combine models is the average. In this way, all datasets are kept within the proprietary organizations, and the only information that gets exchanged is the model parameters, which, in the case of NN, are matrices of floating point numbers representing the weights and the biases associated with the neurons.

Federated Learning performs well when the data is independently and identically distributed (IID) among the involved institutions. Unfortunately, real-world data is often non-IID, and it is well known that this scenario poses critical issues to FL [1]. In a non-IID setting, the data statistics of a single client may be unrepresentative of the global statistics and make the model diverge from the intended solution. Interestingly, Huang et al. show that if the loss surface of the optimization problem is both smooth and convex (which is hardly true in a real scenario), then FedAvg will also converge when the data is non-IID [5].

Recent works have proposed several FL algorithms to cope with non-IIDness problems, such as FedProx [6], FedNova [7], SCAFFOLD [8], and FedCurv [9], which has been tested in [10] and [11]. Notice that all these algorithms are modified versions of FedAvg, and they preserve the principle underneath FedAvg: to average the weights in all the layers of the NN. Most of the common NN architectures employ Batch Normalization (BN) [12], a technique for improving the training of NNs to make them converge faster and to a more stable solution. BN works by standardizing the layers' input for each mini-batch.

In this work, we investigate two aspects of the training FL models, which, differently from the centralized case, happen to be hyper-parameters that can be optimized: the normalization layers and the frequency of model aggregation (epochs per round). We show that the most popular normalization layer (BN) does not couple well with FL for non-IID data and that by substituting BN with alternative normalization FL, a better model can be produced for both the non-IID and IID cases. We also show that building a global model aggregating local models at each epoch is not a good strategy, neither for the quality of the model nor for the execution time. We experiment with two network architectures and five different normalization layers on two public image datasets: MNIST [18] and CIFAR-10 [19].

Results show that the performance of the networks is strictly related to the type of normalization layer adopted.

The main contributions of this work are:

- We provide benchmarks for five different normalization layers: BN, GN, LN, IN, BRN;
- We provide results of experiments on FedAvg on two non-IID settings considering a feature distribution skew and a label distribution skew (in addition to the IID case). To the best of our knowledge, this is the first work providing empirical evidence on the behavior of these normalization layers in common non-IID cases;
- for the most promising normalization layers, we ran extensive tests to discuss how performances are affected by the following factors:
    1) Batch size.
    2) Number of epochs per round (E).
    3) Number of clients.
- We show that choosing the right normalization layer and a suitable number of local gradient descent steps is crucial for obtaining good performances.

This work extends the typical search for optimization of machine learning parameters to federated learning.

The rest of the paper is organized as follows. In Section II, we introduce and discuss recent related works. In Section III, the most used normalization layers are reviewed. In Section IV, the most typical non-IID scenarios are described.

Section V shows and discusses experimental results. Finally, Section VI concludes the paper.

II. RELATED WORK

The main challenges in FL are statistical heterogeneity (non-iidness) and systems heterogeneity (variability of the devices of the federation). In this work, we address the former. In [1], the most common non-IID data settings, that are quantity skew, labels quantity skew (prior shift), feature distribution skew (covariate shift), same label but different features, and same features but different labels, are reviewed. To the best of our knowledge, there are only a few benchmarks for FL dealing with non-IID data. Li et al. in [10] report the analysis of FedAvg [4], FedNova [7], FedProx [6] and SCAFFOLD [8] on nine public image datasets, including MNIST [18] and CIFAR10 [19], split according to three of the previous mentioned non-IID partition strategies, i.e. quantity skew, labels quantity skew and three different versions of feature distribution skew: noise-based, synthetic and real-world feature imbalance. FedAvg [4] is the first proposed FL algorithm. It performs a weighted average of the local models trained on the client's local data. When the batch size is equal to the full local dataset, and the number of epochs per round is equal to one, then FedAvg is better known as FederatedSGD (FedSGD). FedNova adopts the paradigm of FedAvg, but it also normalizes and scales the model's local updates based on the number of local steps. It aims to overcome the problem of objective inconsistency while preserving fast error convergence. FedProx is a re-parametrization of FedAvg in which the size of local updates is restricted by adding an L2 regularization term in the cost function, compelling the local model to stay close to the global model. SCAFFOLD (Stochastic Controlled Averaging for FL) introduces an additional term to the local updates to eliminate the drift caused by local data differences, and it randomly samples a subset of clients to reduce communication overhead. Li et al. [10] show that none of those algorithms outperforms others in all the cases and that non-iidness degrades the performance of FL systems in terms of accuracy, especially in the case of label quantity skew. Another recent work [11] reports an empirical assessment of the behavior of FedAvg and FedCurv [9] on MNIST, CIFAR10 and MedMNIST [20]. Datasets are split according to the same non-IID settings of [10]. FedCurv is an algorithm built on the idea of Continual Learning. It adds the Elastic Weight Consolidation [35] penalty term to the loss function, to minimize the model disparity across the clients of a federation. Authors show that aggregating models at each epoch is not necessarily a good strategy: performing local training for multiple epochs before the aggregation phase can significantly improve performance while also reducing communication costs. FedAvg produced better models in most non-IID settings despite competing with an algorithm explicitly developed to deal with these scenarios (FedCurv).

Results in [11] also confirmed literature sentiment: labels quantity skew and its pathological variant are the most detrimental ones for the algorithms. The same non-IID partitions have already been tested in [24], which proposes a novel technique of non-gradient-descent FL on tabular datasets. Our paper extends [11], deepening the experiments about the number of epochs per round, a hyper-parameter that, if tuned appropriately, can lead to large performance gains. Moreover, we aim to investigate which type of normalization layer better fits FL on non-IID data. Indeed, when data are non-IID, batch statistics do not represent the global statistics, leading NNs equipped with BN to poor results. The most common alternatives to BN are: Group Normalization (GN) [14], Layer Normalization (LN) [15], Instance Normalization (IN) [16] and Batch Renormalization (BRN) [17]. While BN is the most employed normalization method in SOTA algorithms, LN is a fundamental part of the recently proposed Transformer architectures [31] that are becoming widely adopted for solving several learning tasks, such as remote sensing [32], [33] and computer vision [34]. To the best of our knowledge, there are no works benchmarking normalization layers for FL on non-IID data. A previous work [21], proposing a novel form of Transfer Learning through test-time parameters' aggregation, shows that a NN with Batch Normalization [12] does not learn at all, while performance improves only when using Group Normalization [14]. Andreaux et al. propose a novel FL approach by introducing local-statistic BN layers [22]. Their method, called SiloBN, consists in only sharing the learned BN parameters γ and β across clients, while BN statistics μ and σ^2 remain local, allowing the training of a model robust to the heterogeneity of the different centers. SiloBN showed better intra-center generalization capabilities than existing FL methods. FedBN [23] is an FL algorithm that excludes BN layers from the averaging step, outperforming both FedAvg and FedProx in a feature distribution skew setting.

III. NORMALIZATION LAYERS

The majority of the FL algorithms simply apply an aggregation function (like averaging) to all the components of a NN, including weights and biases of the normalization layers. Most of the common NN architectures, like residual networks [13], adopt BN [12] as the normalization layer. However, in contexts like Federated or Transfer Learning, BN may not be the optimal choice, especially when dealing with non-IID data. In this chapter will be reviewed the main characteristics of Batch Normalization and several possible alternatives like Group Normalization (GN) [14], Layer Normalization (LN) [15], Instance Normalization (IN) [16] and Batch Renormalization (BRN) [17].

A. BATCH NORMALIZATION

Batch normalization has recently been extensively adopted by neural networks for their training. The key issue that BN tackles is Internal Covariate Shift (ICS), which is the change in the distribution of the data (or network activations), i.e. the input variables of training and test sets. Informally, at each epoch of training, weights are updated, input data are different, and the algorithm faces difficulties. This results in a slower and more difficult training process because lower learning rates and careful parameter initialization are then required. BN attempts to reduce ICS by normalizing activations to stabilize the mean and variance of the layer's inputs. This accelerates training by allowing the use of higher learning rates and reduces the impact of the initialization. During training, BN normalizes the output of the previous layers along the batch size, height, and width axes to have zero mean and unit variance:

$$\hat{x}_i = \frac{x_i - \mu_m}{\sqrt{\sigma_m^2 + \epsilon}}$$

where x, μ_m and σ_m^2 are respectively the input, the mean, and the variance of a minibatch m, and ϵ is arbitrarily constant greater than zero used for stability in case the denominator is zero. BN also adds two learnable parameters, γ and β that are a scaling and a shifting step, to fix the representation in case the normalization alters what the layer represents: $y_i = \gamma\hat{x}_i + \beta$. Normalized activations will depend on the other samples contained in the minibatch. In the test phase, BN can not calculate statistics; otherwise, it will learn from the test dataset, so it uses the moving averages of minibatch means and variances of the training set. In the case of IID mini-batches, statistical estimations will be accurate if the batch size is large; otherwise, inaccuracies will be compounded with depth, reducing the quality of the models. Non-IID data can have a more detrimental effect on models equipped with BN because batch statistics do not represent global statistics, leading to even worse results. Therefore there is a need to investigate alternatives to BN that can work well with non-IID data and small batch sizes.

B. GROUP NORMALIZATION

Group Normalization is a simple alternative to BN. It divides the channels into different groups and computes within each group the mean μ_i and the variance σ_i along the height and width axes. GN overcomes the constraint on the batch size because it is completely independent of the other input features in the batch, and its accuracy is stable in a wide range of batch size. Indeed, GN has a 10.6% lower error than BN on ResNet-50 [13] trained on ImageNet [14]. The number of groups G is a pre-defined hyperparameter that needs to divide the number of channels C. When G = C, it means that each group contains one channel, and GN becomes Instance Normalization, while when G = 1, it means that one group contains all the channels, and GN becomes Layer Normalization. Both Instance and Layer Normalizations are described below.

C. INSTANCE NORMALIZATION

Instance Normalization is another alternative to BN, firstly proposed for improving NN performances in image generation. It can be seen as a Group Normalization with G = C or as a BN with a batch size of one, so applying the BN formula to each input feature individually. Indeed, IN computes the mean μ_i and the variance σ_i along the height and width axes. As stated before, BN suffers from small batch sizes, so we expect that experiments made with IN will produce worse results than the ones with BN or GN, which can exploit the dependence across the channels.

D. LAYER NORMALIZATION

Layer Normalization was first proposed to stabilize hidden state dynamics on Recurrent Neural Networks (RNNs) [15]. It computes the mean and the variance along the channel, height, and width axes. LN overcomes the constraint on the batch size because it is completely independent of the other input features in the batch. LN performs the same computation both at training and inference times. It can be seen as a GN with G = 1, so with only one group controlling all the channels. As a result, when there are several distributions to be learned among the group of channels, it can perform worse than GN.

E. BATCH RENORMALIZATION

Batch Renormalization [17] is an extension of BN that ensures training and inference models generate the same outputs that depend on individual examples rather than the entire minibatch. BRN is an augmentation of a network that contains batch normalization layers with a per-dimension affine transformation applied to the normalized activations to ensure the match between training and inference models. Reducing the dependence of activation of each sample with other samples in the minibatch can result in a performance increase when data are non-IID.

IV. NON-IID DATA

The most common non-IID data settings are reviewed in [1] that lists five different partitioning strategies: 1) quantity skew, 2) labels quantity skew (prior shift), 3) feature distribution skew (covariate shift), 4) same labels but different features and 5) same features but different labels. In this paper, we consider the same distributions tested in [10] and [11] apart from quantity skew, which is not treated. Indeed, [10] and [11] showed that quantity skew does not hurt the performance of FL models, probably because it results in a different quantity of samples per client, but the distribution of samples is uniform, which is easy to deal with. In this paper, label quantity skew, which is the most detrimental to the FL models' performance, has been extensively tested in a lot of scenarios to show how it is possible to overcome its difficulties. The cases adopted (both IID and non-IID) are briefly described.

- **Uniform Distribution (IID)**: each client of the federation holds the same amount of data, and the distribution is uniform among parties. This is the simplest case for FL algorithms because the distribution is IID.
- **Labels Quantity Skew**: the marginal distributions of labels $P(y_i)$ vary across parties, even if $P(x_i|y_i)$ is the

same. This especially happens when dealing with real FL applications where clients of the federation are distributed among different world regions. Certain data are present only in some countries, leading to the label quantity skew. In this work, we adopted the simplest version of label quantity skew, where each client holds samples belonging to only a fixed amount of classes. In our experiments, we used two as the number of classes per client. Other versions of labels quantity skew can be the Dirichlet labels skew (each client holds samples such that classes are distributed according to the Dirichlet function) and the Pathological labels skew (data are firstly sorted by label and then divided in shards). Some recent works [10], [11] show that the label quantity skew decreases the FL performance by about 15% with respect to the uniform distribution (results on CIFAR10).

- **Feature Distribution Skew**: the marginal distributions $P(x_i)$ vary across parties, even if $P(y|x)$ is shared. This can happen in a lot of ML scenarios; for example, in handwriting recognition, the same words can be written with different styles, stroke widths, and slants. The covariate shift was obtained according to the procedure described in [24]: samples are distributed among clients according to the results of a Principal Component Analysis (PCA) performed on the data. Some recent works [10], [11] show that the covariate shift decreases the FL performance by about 5% with respect to the uniform distribution (results on CIFAR10).

V. EXPERIMENTS

Our experiments have been realized using OpenFL [25], the new framework for FL developed by Intel Internet of Things Group and Intel Labs. OpenFL is a Python 3 library for FL that enables organizations to collaboratively train a model without sharing sensitive information. OpenFL is DL framework-agnostic. Training of statistical models may be done with any deep learning framework, such as TensorFlow or PyTorch, via a plugin mechanism. OpenFL is based on a Director-Envoy workflow which uses long-lived components in a federation to distribute more experiments in the federation. The Director is the central node of the federation. It starts an Aggregator for each experiment, sends data to connected collaborator nodes, and provides updates on the status. The Envoy runs on Collaborator nodes connected to the Director. When the Director starts an experiment, the Envoy starts the Collaborator to train the global model. All the experiments were computed in a distributed environment with ten collaborators. Each collaborator is run on an Intel Xeon CPU (8 cores per CPU), and 1 Tesla T4 GPU. The code used for experimental evaluation is publicly available at https://doi.org/10.5281/zenodo.10380819.

Dataset: We tested FedAvg on MNIST [18] and CIFAR10 [19], that are default benchmarks in ML literature. The details of the datasets are summarized in Table 1. MNIST (Mixed National Institute of Standards and Technology) is a well-known dataset of hand-written digits, including 70.000 grayscale images. Digits, spanning from 0 to 9, have a resolution of 28 × 28 pixels. CIFAR10 (Canadian Institute For Advanced Research) is a widely known dataset in the field of image recognition, encompassing 60.000 RGB images categorized into ten different classes representing real-world objects (like trucks, cars, and aeroplanes). The mage resolution is 32 × 32 pixels. These datasets have been chosen for their popularity in the fields of pattern recognition and computer vision so that the collected results can be compared with the findings of other researchers.

TABLE 1. Statistics of the datasets.

Dataset	Train samples	Test samples	# labels
MNIST	60.000	10.000	10
CIFAR10	50.000	10.000	10

Preprocessing: both datasets were not rescaled: MNIST images are 28×28 while CIFAR10 images are 32×32. As for data augmentation, we performed random horizontal flips and random crops with a probability of 50%. All the datasets were normalized according to their mean and standard deviation.

Model: We employed ResNet-18 [13] and EfficientNet-B0 [26] as classification models, trained by minimizing the cross-entropy loss with mini-batch gradient descent using the Adam optimizer with learning rate 10^{-3}. The local batch size was 128. Both ResNet-18 and EfficientNet-B0 were downloaded from the torchvision.models module. They were originally equipped with BatchNorm. The other different normalization techniques have been hard coded by employing the PyTorch version present in the torch.nn module, adapting them to the number of classes and input channels and substituting them to the original BN layer. We used two networks to show that the results are not model-dependent (See VI for the EfficientNet-B0's results). The scores of baseline models and federated experiments on the uniform and non-IID settings (section. V-A, Tables 2, 4 and 5) are the average (± standard deviation) over five runs. For the extensive experiments on batch size, number of local training steps, and number of clients, we tested only ResNet-18 for only one run.

Normalization Layers: All the normalization layers described before, i.e. Batch Norm, Group Norm, Instance Norm, Layer Norm, and Batch Renormalization, have been applied to the classification model in each experiment. For the most promising normalization, layers have been run to study the impact of the batch size, the number of epochs per round, and the number of clients. For BN, we set the momentum, i.e. the importance given to the previous moving average, to 0.9, according to the SOTA [27] for ResNet-18. For GN, the number of channels must be divisible by the number of groups, so we set the number of groups to 32 for ResNet-18 (one of the possible divisors) and 8 for EfficientNet-B0 (the only possible divisor). All the other normalization layers have been used with their standard PyTorch configuration.

Top-1 accuracy has been employed as a classification metric to compare the performance. Results show the best aggregated model's accuracy. The learning curve of all the experiments can be studied from Figure 1 to Figure 5. Table 2 reports about a non-federated baseline, i.e., the typical AI scenario where the data are centralized. The remaining tables show the performance of FedAvg in different data partitioning scenarios and for different values of some hyperparameters such as batch size, number of epochs per round and number of clients.

A. NORMALIZATION LAYERS AND NON-IID DATA

This subsection presents the results of the three data partitioning scenarios presented: uniform, label quantity skew, and covariate shift. Table 3 shows that normalization levels have a huge impact on the performance of a NN, ranging from very poor levels to almost reaching the level of accuracy in the centralized case. ResNet-18-LN performs slightly better than BN and GN while outperforming IN and BRN in the uniform setting (fig. 1a). In both the labels quantity skew and the covariate shift scenarios, both GN and LN outperform all the other normalization layers; however, they require more training steps to converge, as shown in fig. 1b and fig. 1c. IN does not learn in FL; indeed, since both MNIST and CIFAR10 have ten classes, ResNet-18-IN's performance is like tossing a coin. BRN seems to have a very long learning curve; in fact, it needs a lot of training rounds to reach convergence. However, its performance is still far from the best performances of BN, GN, and LN. For this reason, the following subsections will report results only for the most promising normalization layers: BN, GN, and LN.

B. NORMALIZATION LAYERS AND BATCH SIZE

We examined the effect of a range of batch sizes on training NNs with different normalization layers (Tab. 6 and Fig.2). We trained ResNet-18 on both MNIST and CIFAR10 with a batch size of 8, 16, 32, 64, 128, 256 and 512.

In Tab.6, we can see that GN and LN variants of ResNet-18 consistently outperform BN (batch sizes 8 and 16). In all three variants, the accuracy degrades when the batch size becomes too large (in almost all cases, there is a significant drop in performance when passing batch size 256 to 512). A possible explanation for this phenomenon is that, as stated in [28], ''the lack of generalization ability is due to the fact that large-batch methods tend to converge to sharp minimizers of the training function''. This is especially true in contexts such as FL, where clients have fewer data than in centralized scenarios, and therefore, increasing the batch size has a greater effect.

C. NORMALIZATION LAYERS AND NUMBER OF EPOCHS PER ROUND

The Director-Envoy paradigm provided by OpenFL creates a workspace that allows data scientists to write their own training functions. By default, the number of epochs is equal to 1. Indeed, after every epoch of training, the aggregation strategy chosen is performed (FedAvg). By adjusting this hyper-parameter, it is possible to choose the favourite number of epochs per round before aggregation. We considered two types of experiments to study how accuracy is affected by the number of epochs per round:

- Fix the number of rounds and increase the number of local training steps (Tab.7).
- Fix the number of training epochs to 1000 and vary the ratio of epochs to rounds (Tab. 8).

It can be noted that models benefit from more local steps of gradient descent before doing aggregation. Indeed, accuracy increases as E increases. A possible explanation is that this happens because clients of the federation share a similar loss function shape, and going more and more towards the local minima can be beneficial to reach global optima.

Interestingly, when $E = 1$, BN converges quickly, while GN and LN require more training steps to converge. However, when E increases to 10 or 100, BN also requires more rounds to reach convergence, while the learning curves of GN and LN are unaffected by significant changes.

These results can also be analyzed from a communication point of view: with the same amount of epochs, less communication achieves better results. For example, on CIFAR10, ResNet-GN with $E = 2$ and 500 rounds achieves higher accuracy than ResNet-GN with $E = 1$ and 1000 rounds (Fig.4). This means that perhaps counter-intuitively, training locally before performing aggregation can boost the model's accuracy. This seems to indicate that pursuing local optimizations can lead to better approximations of the local optima. However, at a certain point, increasing E and reducing the number of rounds decreases the performance. This pattern is clearly visible with all the normalization layers and in both datasets. Table 8 shows that we always need an appropriate ratio of epochs to round.

Finally, it has been observed [30] that as the communication frequency diminishes, the ideal settings for local training tend to mirror those of centralized training. For this reason, a more frequent aggregation can require differentiating from the centralized training's optimal parameters.

D. NORMALIZATION LAYERS AND NUMBER OF CLIENTS

A recent work [30] shows that the number of clients of the federation directly affects learning rate and batch size compared to centralized training, affecting thus the global model performance.

We tested the scalability of FL by measuring the effect of the number of clients of the federation, as shown in Fig. 5, and considering two types of experiments:

- a labels quantity skew split of the dataset across a different number of clients (namely 2, 4, 8, and 10). Results are reported in Table 9.
- a uniform dataset split across clients, but considering only some parties. Here the idea is to show how increasing the number of participants, and so the

TABLE 2. Accuracy in the non-federated setting.

Dataset	BN	GN	IN	LN	BRN
MNIST	$99.22\% \pm 0.07\%$	$99.29\% \pm 0.06\%$	$11.36\% \pm 0.00\%$	$\mathbf{99.32\% \pm 0.07\%}$	$99.00\% \pm 0.20\%$
CIFAR10	$\mathbf{82.60\% \pm 0.24\%}$	$81.61\% \pm 0.24\%$	$9.89\% \pm 0.00\%$	$82.06\% \pm 0.40\%$	$81.77\% \pm 0.23\%$

TABLE 3. Accuracy in the uniform setting.

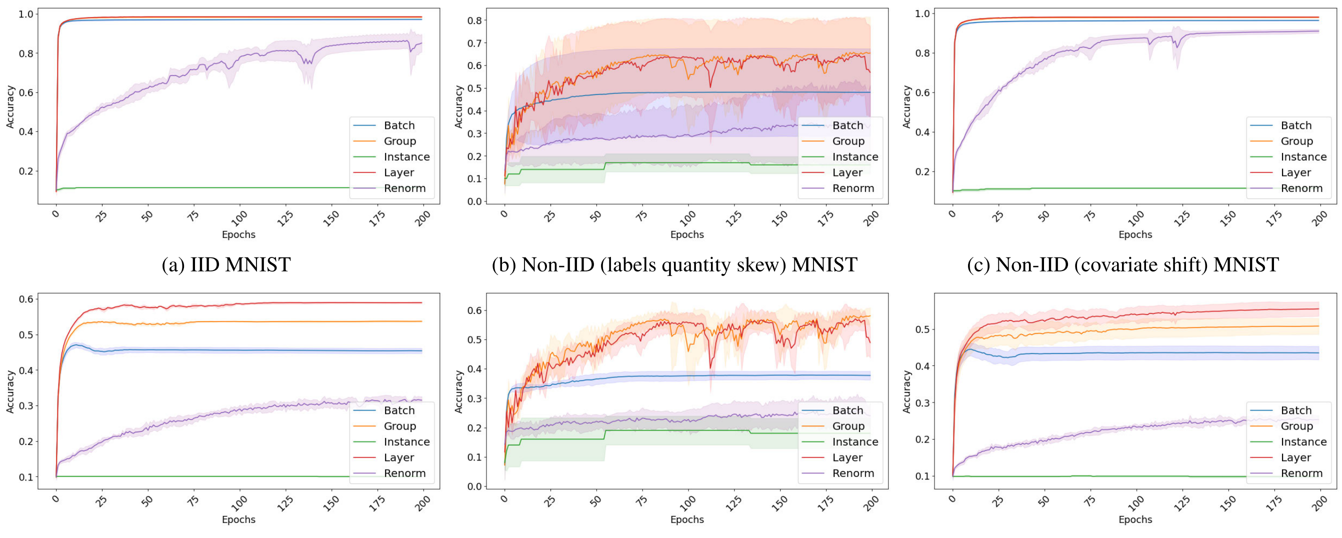
Dataset	BN	GN	IN	LN	BRN
MNIST	$97.12\% \pm 0.24\%$	$98.26\% \pm 0.19\%$	$11.28\% \pm 0.04\%$	$\mathbf{98.51\% \pm 0.06\%}$	$86.54\% \pm 4.24\%$
CIFAR10	$47.12\% \pm 0.82\%$	$53.99\% \pm 0.23\%$	$10.04\% \pm 0.07\%$	$\mathbf{59.06\% \pm 0.25\%}$	$32.36\% \pm 1.07\%$

TABLE 4. Accuracy in the labels quantity skew setting.

Dataset	BN	GN	IN	LN	BRN
MNIST	$82.25\% \pm 7.70\%$	$97.32\% \pm 0.46\%$	$18.53\% \pm 4.18\%$	$\mathbf{97.68\% \pm 0.22\%}$	$74.66\% \pm 3.20\%$
CIFAR10	$38.02\% \pm 1.64\%$	$\mathbf{58.91\% \pm 3.10\%}$	$19.00\% \pm 5.47\%$	$57.98\% \pm 3.88\%$	$27.34\% \pm 5.56\%$

TABLE 5. Accuracy in the covariate shift setting.

Dataset	BN	GN	IN	LN	BRN
MNIST	$96.42\% \pm 0.25\%$	$\mathbf{97.96\% \pm 0.19\%}$	$11.51\% \pm 0.23\%$	$97.37 \pm 1.63\%$	$91.01\% \pm 1.12\%$
CIFAR10	$44.58\% \pm 1.96\%$	$51.04\% \pm 2.83\%$	$10.08\% \pm 0.23\%$	$\mathbf{55.54 \pm 2.22\%}$	$26.20\% \pm 1.48\%$



(a) IID MNIST (b) Non-IID (labels quantity skew) MNIST (c) Non-IID (covariate shift) MNIST

(d) IID CIFAR10 (e) Non-IID (labels quantity skew) CIFAR10 (f) Non-IID (covariate shift) CIFAR10

FIGURE 1. Accuracies of ResNet-18 on the uniform and non-IID cases. BN, GN, and LN require a few rounds to reach convergence in the uniform setting, while they need more training steps to converge in non-IID scenarios. It is clearly shown that BRN requires a very long learning curve and that IN does not learn in FL.

quantity of data, can be beneficial to the federation. Results are reported in Table 10.

We can observe (Table 9) that the accuracy significantly increases when decreasing the number of clients. Indeed, when the number of parties is small, the amount of local data increases, leading to better local models, and aggregating fewer models can result in less information loss. Moreover, we can note the importance of normalization layers in FL: GN and LN variants of ResNet-18 in the ten-client scenario perform better than BN in a two-client scenario on CIFAR10, while on MNIST, there is only a slight drop.

Table 10 shows the results in an IID scenario considering only some shards of the dataset. In this case, the amount of local data remains the same in each configuration; however, the federation's total amount of data varies according to the

TABLE 6. Accuracy in the labels quantity skew setting as the batch size varies.

Dataset	Batch size	BN	GN	LN
MNIST	8	94.77%	97.85%	98.32%
	16	81.02%	97.92%	98.83%
	32	98.00%	97.93%	98.36%
	64	73.89%	95.91%	98.13%
	128	87.07%	96.84%	97.39%
	256	90.33%	97.22%	97.39%
	512	87.86%	97.97%	96.99%
CIFAR10	8	40.23%	59.95%	66.63%
	16	45.41%	61.87%	56.42%
	32	42.26%	55.61%	52.70%
	64	43.02%	59.43%	55.07%
	128	38.02%	60.17%	60.36%
	256	46.69%	56.18%	56.92%
	512	35.86%	51.92%	59.73%

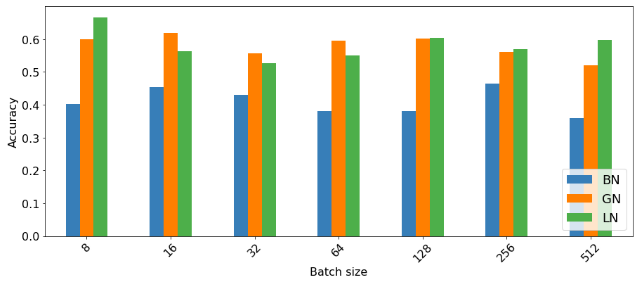


FIGURE 2. Accuracies on CIFAR10 and different batch sizes. Accuracy degrades when the batch size becomes too large.

TABLE 7. Accuracy in the labels quantity skew setting as the number of epochs per round varies.

Dataset	Epochs	BN	GN	LN
MNIST	1	87.07%	96.84%	97.39%
	10	79.82%	98.83%	98.95%
	100	92.88%	97.83%	99.04%
CIFAR10	1	38.02%	60.17%	60.36%
	10	55.53%	54.89%	63.68%
	100	54.19%	73.32%	68.40%

number of parties. Increasing the quantity of data in the federation by increasing the number of clients benefits the aggregated model.

E. ANALYSIS OF COMPUTATION TIME

Table 14 in the Appendix section summarizes the wall-clock times (mean of five different runs) required for training the models for 200 federation rounds on CIFAR10. Results, expressed in the HH:MM time format, reveal an easily discernible pattern. All the alternatives to BN decrease the wall-clock time by about 20%. While the wall-clock times are the combination of the computation (train and test phases) and communication (exchanging parameters between the clients and the central server) times, it is important to note that the overhead introduced by BN, with respect to the alternative normalization layers, highly depends on the computation phase. Indeed, normalization introduces a small number of learnable parameters for every layer of the model. So, the increase in magnitude (in MB) of the weights exchanged between the federation's participants is negligible. In particular, compared with its alternatives, the possible overhead introduced by the saving and communication of BN's moving averages is still negligible. Finally, the main slowdown introduced by BN should be due to the steps

TABLE 8. Comparison between different epochs per round in labels quantity skew setting.

Dataset	Epochs	Rounds	BN	GN	LN
MNIST	1	1000	91.15%	98.92%	98.77%
	2	500	84.68%	98.85%	99.05%
	5	200	92.42%	98.33%	99.12%
	10	100	79.40%	98.46%	98.69%
	20	50	77.86%	97.17%	97.26%
	50	20	54.18%	79.77%	74.40%
	100	10	60.29%	78.02%	78.86%
CIFAR10	1	1000	43.12%	56.94%	57.03%
	2	500	50.09%	65.14%	62.63%
	5	200	47.85%	63.68%	64.56%
	10	100	51.37%	51.23%	61.22%
	20	50	48.16%	54.33%	65.36%
	50	20	38.92%	56.53%	52.67%
	100	10	33.06%	48.24%	50.56%

TABLE 9. Accuracy in the labels quantity skew setting as the number of collaborators varies.

Dataset	Clients	BN	GN	LN
MNIST	2	99.02%	99.61%	99.85%
	4	91.85%	99.47%	98.36%
	8	88.38%	96.49%	97.68%
	10	87.07%	96.84%	97.39%
CIFAR10	2	44.57%	74.15%	75.63%
	4	55.30%	50.50%	67.69%
	8	51.27%	60.75%	65.00%
	10	38.02%	60.17%	60.36%

TABLE 10. Accuracy in the *i.i.d.* setting using only some shards of the dataset.

Dataset	Clients	BN	GN	LN
MNIST	2	94.64%	96.26%	97.38%
	4	96.18%	97.52%	98.10%
	8	96.79%	97.98%	98.34%
	10	97.12%	98.26%	98.51%
CIFAR10	2	41.80%	45.37%	51.73%
	4	43.47%	48.74%	54.55%
	8	45.81%	53.18%	58.36%
	10	47.12%	53.99%	59.06%

TABLE 11. Accuracy in the uniform setting with EfficientNet-B0.

Dataset	BN	GN	IN	LN	BRN
MNIST	65.09%	95.07%	11.26%	96.12%	11.02%
CIFAR10	29.88%	29.93%	10.08%	38.98%	10.48%

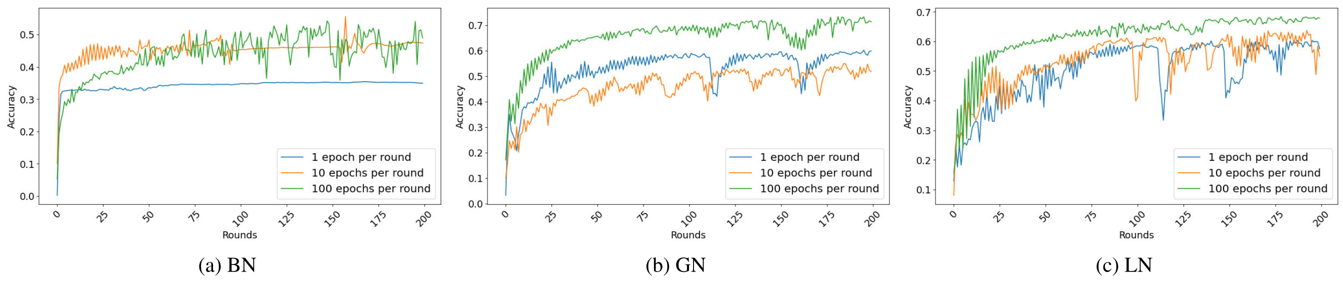


FIGURE 3. Accuracies on CIFAR10 and different epochs per round. Accuracy increases as BN converges and lates as E increases, while GN and LN follow an inverse pattern.

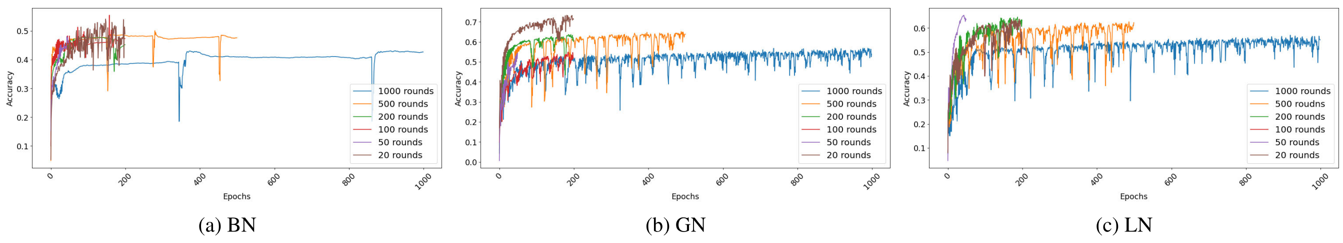


FIGURE 4. Accuracies on CIFAR10 fixing the number of epochs to 1000 and varying the ratio of epochs to rounds.

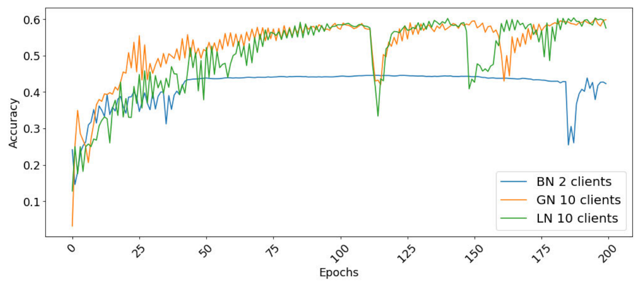


FIGURE 5. Accuracies on CIFAR10 and different number of clients. Accuracy of ResNet-18-BN on 2 parties is still lower than ResNet-18-GN/LN on 10 parties.

TABLE 12. Accuracy in the labels quantity skew setting with EfficientNet-B0.

Dataset	BN	GN	IN	LN	BRN
MNIST	55.83%	91.91%	24.13%	94.35%	45.69%
CIFAR10	29.98%	27.53%	14.92%	37.71%	22.61%

of synchronization among mini-batches. Indeed, in BN, the local averages and standard deviations have to be combined into a single global mean and variance. This synchronization overhead is not present in other normalization techniques computing statistics per each sample individually (LN), per group (GN, IN). Surprisingly, while BRN exploits the same mechanism of statistics computation but a slightly different normalization step from BN, its wall-clock times are lower than BN's. This behavior will be further investigated for future work.

TABLE 13. Accuracy in the covariate shift setting with EfficientNet-B0.

Dataset	BN	GN	IN	LN	BRN
MNIST	65.10%	94.86%	11.19%	95.81%	61.90%
CIFAR10	26.78%	28.01%	10.16%	48.10%	10.01%

TABLE 14. Wall-clock execution times [HH:MM] to train models on CIFAR10 for 200 federated training rounds of 1 epoch.

Distribution	BN	GN	IN	LN	BRN
UNIFORM	04:43	03:48	03:49	03:49	03:54
LABEL QUANTITY SKEW	04:42	03:53	03:51	03:53	03:56
COVARIATE SHIFT	04:47	03:52	03:54	03:49	03:57

VI. CONCLUSION

This work aims to improve the effectiveness of federated learning, focusing on hyper-parameter optimization, starting from understanding which hyper-parameters specifically affect the training of a federated model differently from centralized training. We specifically focused on layer normalization, which is also a hyper-parameter of centralized training, and frequency of model aggregation, which is not an issue in centralized training.

We experimented with two network architectures and five normalization layers on two public image datasets. We tested Batch, Group, Instance, Layer Normalization, and Batch Renormalization in the uniform, label quantity skew, and covariate shift settings. Although BN is the SOTA for classical ML techniques, in our experiments, GN and LN outperformed the other normalization layers in all the FL partitioning strategies.

This paper provides a benchmark for the most common normalization layers of NNs, helping researchers to compare and contrast their findings with those of other scientists. As a drawback, we can underline that BN is the most employed solution for SOTA algorithms. This can make it difficult and unfair to compare newly proposed solutions that adopt different normalization layers and SOTA methods.

Through extensive experimentation, we analyzed how the batch size, the number of epochs per round, the number of rounds, and the number of clients of the federation affect the aggregated model performance. These additional tests have been conducted in the labels quantity skew scenario, which is the most challenging for FL algorithms, considering the best three normalization layers: BN, GN, and LN.

GN and LN outperform BN in almost all the tests. Results show that regardless of the batch size, GN and LN consistently outperform BN, although batch size affects the model's performance in all cases. Unexpectedly, we observed that the plot of the quality of the model against the frequency of model aggregation (epochs per round) consistently exhibits a maximum at a few epochs per round. For FL, the number of epochs per round exhibits similar behavior of batch size for centralized training.

Eventually, we tested the scalability of FL systems. We noted that FL is not scalable under a strong scalability assumption, i.e., increasing the number of clients while maintaining the size of local datasets constant. However, GN and LN on ten clients still outperform BN on two clients. The scalability has also been tested in the IID scenario under the weak scalability assumption, i.e., increasing the number of clients while maintaining the size of the local dataset per client constant. In this case, the federation's data changes with the number of clients, and the model's performance increases with the number of parties.

In future work, considering the close relationship between the communication frequency, the batch size, and the number of workers, we aim to understand deeper if a higher learning rate and a reduced batch size are required when increasing the number of epochs per round. Indeed, a smaller batch size may be required in order to have more model updates, together with a higher learning rate, which can confirm that with more epochs per round, the global model parameters have to go away from centralized optimal parameters [30].

Proposing a new normalization scheme specifically devised for FL on non-IID data is another interesting direction for future work. Taking inspiration from the parallel tempering method, we plan to implement a mean-exchange algorithm in which the weights of the normalization layers are exchanged among the federation's parties rather than averaged.

APPENDIX

Accuracies on the uniform and non-iid data settings using and EfficientNet-B0 [26].

REFERENCES

[1] P. Kairouz, "Advances and open problems in federated learning," *Found. Trends Mach. Learn.*, vol. 14, nos. 1–2, pp. 1–210, 2021.

[2] S. Callaghan, "Data sharing in a time of pandemic," *Patterns*, vol. 1, no. 5, Aug. 2020, Art. no. 100086.

[3] P. Voigt and A. Bussche, *The EU General Data Protection Regulation (GDPR): A Practical Guide*. Cham, Switzerland: Springer, 2017.

[4] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. Int. Conf. Artif. Intell. Statist. (AISTATS)*, Apr. 2017, pp. 20–22.

[5] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of FedAvg on non-IID data," 2019, *arXiv:1907.02189*.

[6] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," *Proc. Mach. Learn. Syst.*, vol. 2, pp. 429–450, Mar. 2020.

[7] J. Wang, Q. Liu, H. Liang, G. Joshi, and H. V. Poor, "Tackling the objective inconsistency problem in heterogeneous federated optimization," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 7611–7623.

[8] S. P. Karimireddy, S. Kale, and M. Mohri, "SCAFFOLD: Stochastic controlled averaging for federated learning," in *Proc. 37th Intl. Conf. Mach. Learn.*, Jul. 2020, pp. 13–18.

[9] N. Shoham, T. Avidor, A. Keren, N. Israel, D. Benditkis, L. Mor-Yosef, and I. Zeitak, "Overcoming forgetting in federated learning on non-IID data," 2019, *arXiv:1910.07796*.

[10] Q. Li, Y. Diao, Q. Chen, and B. He, "Federated learning on non-IID data silos: An experimental study," in *Proc. IEEE 38th Int. Conf. Data Eng. (ICDE)*, Kuala Lumpur, Malaysia, May 2022, pp. 965–978.

[11] B. Casella, R. Esposito, C. Cavazzoni, and M. Aldinucci, "Benchmarking FedAvg and FedCurv for image classification tasks," in *Proc. 1st Italian Conf. Big Data Data Sci.*, 2022, pp. 1–12.

[12] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. 32nd Int. Conf. Mach. Learn. (ICML)*, Lille, France, Jul. 2015, pp. 6–11.

[13] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[14] Y. Wu and K. He, "Group normalization," in *Proc. 15th Eur. Conf.*, 2018, pp. 3–19.

[15] J. Lei Ba, J. Ryan Kiros, and G. E. Hinton, "Layer normalization," 2016, *arXiv:1607.06450*.

[16] D. Ulyanov, A. Vedaldi, and V. Lempitsky, "Instance normalization: The missing ingredient for fast stylization," 2016, *arXiv:1607.08022*.

[17] S. Ioffe, "Batch renormalization: Towards reducing minibatch dependence in batch-normalized models," in *Proc. NIPS*, Long Beach, CA, USA, Dec. 2017, pp. 1945–1953.

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

[19] A. Krizhevsky, *Learning Multiple Layers of Features From Tiny Images*. Princeton, NJ, USA: Citeseer, 2009.

[20] J. Yang, R. Shi, D. Wei, Z. Liu, L. Zhao, B. Ke, H. Pfister, and B. Ni, "MedMNIST V2—A large-scale lightweight benchmark for 2D and 3D biomedical image classification," 2021, *arXiv:2110.14795*.

[21] B. Casella, A. Chisari, S. Battiato, and M. Giuffrida, "Transfer learning via test-time neural networks aggregation," in *Proc. 17th Int. Joint Conf. Comput. Vis., Imag. Comput. Graph. Theory Appl.*, 2022, pp. 642–649.

[22] M. Andreux, J. O. du Terrail, C. Beguier, and E. W. Tramel, "Siloed federated learning for multi-centric histopathology datasets," in *Proc. Domain Adaptation Represent. Transf., Distrib. Collaborative Learn. 2nd MICCAI Workshop*, 2020, pp. 129–139.

[23] X. Li, M. Jiang, X. Zhang, M. Kamp, and Q. Dou, "FedBN: Federated learning on non-IID features via local batch normalization," in *Proc. 9th Int. Conf. On Learn. Represent.*, May 2021, pp. 1–27.

[24] M. Polato, R. Esposito, and M. Aldinucci, "Boosting the federation: Cross-silo federated learning without gradient descent," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Padua, Italy, Jul. 2022, pp. 1–10.

[25] G. Anthony Reina, A. Gruzdev, P. Foley, O. Perepelkina, M. Sharma, I. Davidyuk, I. Trushkin, M. Radionov, A. Mokrov, D. Agapov, J. Martin, B. Edwards, M. J. Sheller, S. Pati, P. Narayana Moorthy, S.-H. Wang, P. Shah, and S. Bakas, "OpenFL: An open-source framework for federated learning," 2021, *arXiv:2105.06413*.

[26] M. Tan and Q. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *Proc. 36th Int. Conf. Mach. Learn.*, vol. 97, Jun. 2019, pp. 9–15.

[27] T. Moreau et al., "Benchopt: Reproducible, efficient and collaborative optimization benchmarks," 2022, *arXiv:2206.13424*.

[28] N. S. Keskar, J. Nocedal, P. T. P. Tang, D. Mudigere, M. Smelyanskiy, "On large-batch training for deep learning: Generalization gap and sharp minima," in *Proc. 5th Int. Conf. On Learn. Represent.*, 2017, pp. 1–26.

[29] Y. Arfat, G. Mittone, I. Colonnelli, F. D'Ascenzo, R. Esposito, and M. Aldinucci, "Pooling critical datasets with federated learning," in *Proc. 31st Euromicro Int. Conf. Parallel, Distrib. Netw.-Based Process. (PDP)*, Mar. 2023, pp. 329–337.

[30] M. Kamp, L. Adilova, J. Sicking, F. Hüger, P. Schlicht, T. Wirtz, and S. Wrobel, "Efficient decentralized deep learning by dynamic model averaging," in *Proc. Mach. Learn. And Knowl. Discovery Databases—Eur. Conf.*, 2018, pp. 393–409.

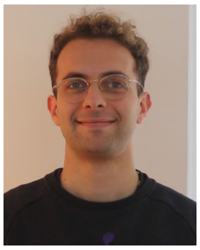
[31] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. Adv. Neural Inform. Process. Syst. (NIPS)*, 2017, pp. 5998–6008.

[32] Y. Xiao, Q. Yuan, K. Jiang, J. He, C.-W. Lin, and L. Zhang, "TTST: A Top-k token selective transformer for remote sensing image super-resolution," *IEEE Trans. Image Process.*, vol. 33, pp. 738–752, 2024, doi: 10.1109/TIP.2023.3349004.

[33] Y. Xiao, Q. Yuan, K. Jiang, X. Jin, J. He, L. Zhang, and C.-W. Lin, "Local-global temporal difference learning for satellite video super-resolution," *IEEE Trans. Circuits Syst. Video Technol.*, Jun. 2024, doi: 10.1109/TCSVT.2023.3312321.

[34] K. Jiang, Z. Wang, C. Chen, Z. Wang, L. Cui, and C.-W. Lin, "Magic ELF: Image deraining meets association learning and transformer," in *Proc. 30th ACM Int. Conf. Multimedia*, Lisboa, Portugal, Oct. 2022, pp. 827–836, doi: 10.1145/3503161.3547760.

[35] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, and R. Hadsell, "Overcoming catastrophic forgetting in neural networks," 2016, *arXiv:1612.00796*.



BRUNO CASELLA received the B.S. degree in computer engineering and the M.Sc. degree in data science for management from the University of Catania, in 2020 and 2021, respectively. He is currently pursuing the Ph.D. degree in modeling and data science with the University of Turin. His current research interests include federated learning, transfer learning, and distributed computing.

ROBERTO ESPOSITO received the Ph.D. degree in computer science from the University of Turin, in 2003. He is currently an Associate Professor with the University of Turin

ANTONIO SCIARAPPA received the Ph.D. degree in theoretical particle physics from SISSA, in 2015. He is currently a Researcher with the Leonardo Labs, Genoa, Italy. His current research interests include parallel and distributed computing.

CARLO CAVAZZONI received the Ph.D. degree in computational physics from SISSA. He is currently the Senior Vice President of Cloud Computing and the Head of the High-Performance Computing of Leonardo S.p.A

MARCO ALDINUCCI received the Ph.D. degree from the University of Pisa, in 2003. He is currently a Full Professor and a Principal Investigator (PI) of the Parallel Computing Research Group, University of Turin. He has been a Researcher with Italian National Research Council (CNR).

Open Access funding provided by 'Università degli Studi di Torino' within the CRUI CARE Agreement