FEDBS: Learning on Non-IID Data in Federated Learning using Batch Normalization

1st Meryem Janati Idrissi
SCCS
Mohammed VI Polytechnic University
Benguerir, Morocco
Meryem.Janati@um6p.ma

2nd Ismail Berrada
SCCS
Mohammed VI Polytechnic University
Benguerir, Morocco
Ismail.Berrada@um6p.ma

3rd Guevara Noubir
Northeastern University
Boston, MA
G.Noubir@northeastern.edu

***Abstract*—Federated learning (FL) is a well-established distributed machine-learning paradigm that enables training global models on massively distributed data i.e., training on multi-owner data. However, classic FL algorithms, such as Federated Averaging (*FedAvg*), generally underperform when faced with Non-Independent and Identically Distributed (Non-IID) data. Such a problem is aggravated for some hyperparametric methods such as optimizers, regularization, and normalization techniques. In this paper, we introduce *FedBS*, a new efficient strategy to handle global models having batch normalization layers, in the presence of Non-IID data. *FedBS* modifies *FedAvg* by introducing a new aggregation rule at the server-side, while also retaining full compatibility with Batch Normalization (BN). Through our evaluations, we have empirically proven that *FedBS* outperforms both classical FedAvg, as well as the state-of-the-art *FedProx* through a comprehensive set of experiments conducted on Cifar-10, Mnist, and Fashion-Mnist datasets under various Non-IID data settings. Furthermore, we observed that in some cases, *FedBS* can be 2× faster than other FL approaches, coupled with higher testing accuracy.**

***Index Terms*—Federated learning, Batch Normalization, Non-IID data.**

I. Introduction

Smartphones, wearable sensors and autonomous vehicles generate a substantial amount of valuable data daily [1]. This large-scale data have become crucial for service providers in order to improve AI based products and user experience via training high-performance machine learning models. Meanwhile, the collected data might contain private sensitive users' information, which should not be publicly disclosed, hence sharing it with a centralized entity is highly discouraged and raises both privacy and security concerns.

Federated Learning (FL) was firstly introduced in 2016, by McMahan et al. [2], as a decentralized machine learning framework where multiple entities jointly train a global statistical model $f(w)$ without the need to send the local data to a centralized server. At each communication round of the training, a fraction of the clients are chosen to collaboratively minimize an objective function $F(w)$:

$$\underset{w}{min}\ F(w) = \underset{w}{min} \sum_{k=1}^{N} p_k F_k(w)$$

where $F_k(w) = \frac{1}{n_k} \sum_{j_k=1}^{n_k} f_{j_k}(w; x_{j_k}, y_{j_k})$ is the local objective function measuring the empirical risk over local data, N is the number of entities, n_k is the number of data points available at the entity k, and $p_k = \frac{n_k}{n}$ is the relative impact of each entity such that $n = \sum_k n_k$ is the total number of data samples.

FL is regarded as a seminal work and was adopted by many applications [3], [4]. However, FL has induced three main challenges:

1) Communication bottleneck [5]–[7]: Although the computation power of mobile devices has increased significantly in the last decade, the bandwidth of wireless communications has not improved greatly. The bottleneck has then shifted from computation into communication in which the limited computation bandwidth has significantly slowed down the convergence time.
2) Systems heterogeneity [8]: The data-owners participating in FL process generally vary in terms of system-level characteristics (i.e. storage, computational power, communication bandwidth, etc) which raises issues such as straggler mitigation and fault tolerance.
3) Statistical heterogeneity [9], [10]: Intuitively, each device in the FL setting, has local data samples that are biased by the unique device user environment and characteristics. Therefore, in practice, local data on edge devices is not always IID (Independent and Identically Distributed). As a result, the IID assumptions are violated in FL and common architectures tend to perform poorly in this case, for instance Batch Normalization (BN) as we will show in further details in section III.

Since its first introduction in 2015 [11], BN is one of the most widely used architectures in modern deep learning models. Its benefits range from model stability to faster convergence and robustness to hyperparameters. BN normalizes the input layer by adjusting and scaling the activations. During training, for each mini-batch, the normalization layer converts the input to have zero mean and unit variance using the local statistics of the device's data. For the evaluation, running estimates of mean and variance are used for normalization. Thus, when the data is Non-IID, the results of the evaluation differ from training.

In this work, we addressed the challenges previously highlighted of Non-IID data and their impact on models trained with batch normalization layers. First, by comparatively discussing the different alternatives to batch normalization techniques, then we propose *FedBS*, a new generic aggregation strategy to handle batch statistics in FL with Non-IID settings. The intuition behind *FedBS* is to warm up the global model in the early rounds by weighting the local model of each client by the model itself instead of the local data size. In other words, the weight of each client during the aggregation step is calculated based on the local loss, the larger the loss the larger the weight of the corresponding client. When all clients start to converge to the global model and their loss is close, we switch to equal weights for all clients and use *FedProx*. Our work's main contributions are as follows:

- Highlighting the ubiquity of BN layers in deep learning applications and the issues faced when using BN with Non-IID data in FL.
- Introducing *FedBS* as a novel FL approach leveraging local batch statistics when using deep learning models with normalization layers in FL. *FedBS* modifies the naive way of computing the weight vector of the local models based on the data size only.
- Empirical based results showing that our proposed approach outperforms state-of-the-art methods on both a variety of real world datasets and also under various Non-IID settings.

To the best of our knowledge, our work is the first to propose a new aggregation framework for batch statistics in FL at the server side, and also outperforms state-of-the-art approaches.

II. Related Works

Most decentralized machine learning algorithms are envisioned with IID data in mind. However, in real world situations, the data is commonly generated in different contexts, time windows, and locations. Thus, the IID assumption does not hold and distribution of data stored across edge devices is not drawn from some global distribution and hence not representative of the overall distribution. Although McMahan et al. [2] have shown the robustness of *FedAvg* towards certain Non-IID distributions, several studies have demonstrated the divergence and instability of *FedAvg* in such setting. For instance, the experiments presented in [12] have shown a significant performance degradation coupled with a communication cost of *FedAvg* with a highly skewed Non-IID data where the accuracy of neural networks was reduced up to 11% for MNIST, 51% for CIFAR-10 and 55% for keyword spotting (KWS) datasets. Similarly, Li et al. [13] have analyzed the convergence of *FedAvg* in a more realistic setting where the data is Non-IID and the device participation rate is low in each round. Both theoretical studies and empirical results have shown that the convergence rate is gravely hindered in such circumstances.

Several subsequent works have addressed these challenges under some assumptions. Zhao et al. [12] proposed an enhanced version of *FedAvg* based on a data-sharing approach. The server holds a global dataset G drawn from a uniform distribution and shares a random small portion of G with all clients. Additionally, instead of starting with a global model initialized randomly, the server warms up the model centrally on the server-side proxy data. However, this strategy requires some public dataset being available for a particular task which might be unrealistic in practice. A similar approach [14] suggested the use of Federated Augmentations where each device can generate the missing samples using a generative model. However, this requires sharing some devices' data samples with the server, which violates the key privacy goal of FL. Li et al. [15] have proposed *FedProx*, a generalized version of *FedAvg* to tackle statistical heterogeneity. *FedProx* modifies *FedAvg* by adding a regularization term to the local objective functions in order to limit the impact of the local updates and thus restraining the local models divergence.

In another line of work, [16]–[19] allow a partial client participation based on a conditional client selection. In particular, [16] studies the convergence of *FedAvg* under biased client selection and proposes non-uniform clients' sampling based on their losses. The proposed strategy shows faster convergence and higher testing accuracy. However, such approaches are often not governed by the server but by the client availability.

The issues with Non-IID data become even more challenging when using some hyperparameters under Non-IID setting that showed their success under IID setting. [20] discussed the problem of skewed label partitions and have shown the vulnerability of deep neural networks trained with Batch Normalization under Non-IID setting in a decentralized environment.

Sharing the same concern as we do, *SiloBN* [21] addresses the issue of BN in FL by modifying how the synchronization of BN parameters is made. *SiloBN* proposes sharing only the trainable parameters γ and β with the server while keeping BN statistics local. On the other hand, [22] have proposed *FedBN* which unlike SiloBN keeps all BN parameters at the clients and updates them locally without communication with the server. The empirical results presented in the paper have shown that *FedBN* performs better than standard *FedAvg*. However, the paper does not study the efficiency of *FedBN* in the case of unbalanced labels and for more complex datasets, moreover, the paper does not cover the case where the testing clients have no data to compute BN statistics on.

III. Background

In this section, we first introduce the basic concepts related to N on-IID data, then we review the different techniques used in deep learning to normalize the data.

A. Non-IID Settings

Let us consider a learning task $\mathcal{T}$ with features x and labels y. $\mathcal{P}_i$ is the data distribution of the client i. In Non-IID settings, the difference between two distributions $\mathcal{P}_i$ and $\mathcal{P}_j$ for different clients i and j can be classified into various classes [23]:

Violation of Identicalness

- *Feature distribution skew:* The marginal distributions $\mathcal{P}_i(x)$ may vary across clients. The input features are not evenly distributed between clients. For instance, in a handwriting recognition task, users write the same words in a different way.
- *Same label, different features (concept shift):* The conditional distributions $\mathcal{P}_i(x|y)$ may vary across clients. Different features across different clients might be labeled with the same label. This is due to cultural differences and standards of living, etc. For example, in some regions, "Panda" is called "Panda" and in other regions, "Red Panda" is also called "Panda".
- *Same features, different labels (concept shift):* The conditional distributions $\mathcal{P}_i(y|x)$ may vary across clients. In this case, the same features across different clients are labeled differently. For instance, "Leopards" can either be called Leopards or "Pards".
- *Unbalancedness:* Different clients can hold vastly different amount of data.

Violation of Independence

- *Inter-partition correlation:* Existing data on a client might be dependent on other existing data on another client. For example, a phenomenon to be fully understood, all the information about the different devices are needed.
- *Intra-partition correlation:* Data samples hold by a single partition are dependent. For example, the consecutive frames in a video are highly correlated.

B. Batch Normalization

BN has become an indispensable technique in deep learning and it is one of the components that has helped boosting machine learning applications in the last few years. In order to stabilize and speed up neural network training, BN normalizes the input distribution to have a mean of zero and a variance of one. BN acts differently between the training and inference phases. During training, BN uses mini-batch means and variances for normalization. In fact, for each mini-batch B, BN first computes the mean μ_B and the variance σ_B, and then normalizes each input x_i in B as follows:

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

where ϵ is a small constant added for numerical stability. Further, BN adds two learnable parameters, namely the γ coefficient and the β bias, to automatically scale and shift the normalized pre-activations. In other words, these parameters allow the network to convert the mean and variance to their optimal values:

$$y_i = \gamma \hat{x}_i + \beta$$

At the inference time, as each sample is processed separately, computing the mini-batch mean and variance is infeasible. Thus, during the training, we estimate the mean and variance over the whole dataset as follows:

$$E = \frac{1}{m}\sum_{i=1}^{j} \mu_B^{(i)} \qquad Var = (\frac{m}{m-1})\frac{1}{m}\sum_{i=1}^{j} \sigma_B^{2^{(i)}}$$

where m is the mini-batch size and j is the total number of mini-batches. These estimated values are then used at the inference time as follows:

$$y = \frac{\gamma}{\sqrt{Var + \epsilon}} x + (\beta + \frac{\gamma E}{\sqrt{Var + \epsilon}})$$

C. Non-IID and Batch Normalization

Generally, Non-IID data have not been an obstacle towards learning from distributed data since data centers have access to the entire dataset and can distribute the data according to IID assumption. However, in FL, the server does not have such a luxury, as it does not have access to local client data. In real-world scenarios, each client may correspond to an end device, a particular user or a geographic location. Thus, the data available locally are more likely to be dependent and non-identical. Each partition has a different probability distribution that fails to represent the population distribution. For example, data collected from mobile devices will reflect the preferences of each user based on his age, location, gender, etc. The difference in local distributions make learning from local data a challenging task due to the inconsistency between 1) the results of the gradient algorithm executed locally on each client k that aims to minimize a local objective function $F_k(w)$ over m_k samples and 2) the global objective of minimizing the global function $F(w)$ over all data samples $\sum_k m_k$.

It is known that BN becomes ineffective in certain settings (small and Non-IID mini-batches [24]). In fact, if μ_B and σ_B diverge from the global mean and variance of the whole dataset, then the testing accuracy will be unsurprisingly low. In FL settings, where each data partition could differ from the other, the performance degradation is more severe as μ_B and σ_B vary significantly across partitions.

D. Alternatives to Batch Normalization

Some existing techniques can be used as alternatives to BN when processing Non-IID data:

1) Group Normalization (GN):: GN [25] might be an alternative to BN, e.g. when the batches are too small (due to memory limitations) to derive accurate statistics. GN divides the channels of each sample into groups and computes the mean and variance over these groups. Hence, GN is independent of mini-batches and unlike BN, the process of normalization is the same for training and inference phases.

2) Batch Renormalization (BRN):: BRN [24] attempts to rectify the difference in activities between training and inference due to the difference in normalization between the two phases. BRN introduces an additional affine transformation from batch statistics (i.e. μ_B and σ_B) to the running statistics (μ and σ):

$$\hat{x}_i = \frac{x_i - \mu_B}{\sigma_B}.r + d$$

where $r = \frac{\sigma_B}{\mu}$ and $d = \frac{\mu_B - \mu}{\sigma}$. This transformation ensures that the effect of previous normalization parameters is taken into account. In other words, the parameters r and d add the impact that relates the statistics computed during the training and those during the inference phase. BRN can be seen as a simple generalization of BN (i.e. by simply putting $r = 1$ and $d = 0$, we revert to BN).

3) Fixed Update Initialization (FixUp):: FixUp [26] is an initialization technique for Residual Networks. It suggests that normalization is not necessary to achieve state-of-the-art performance and explores the effect of initialization and normalization in residual networks. By properly rescaling the weights initialization coupled with a proper regularization, we can train deep residual networks while avoiding exploding/vanishing gradients.

IV. Our Approach: FedBS

In the case of FL with IID settings, training neural networks with batch normalization layers does not pose any problem since each local data is drawn from the global distribution, and therefore the mini-batch statistics (μ and σ^2) are representative of the whole dataset statistics. However, in Non-IID setting, the statistics calculated by each client are biased to its local data and vary across clients.

In spite of previously discussed alternatives showing some promising results which helped maintain the performance of BN in a few cases [20], they come with some side effects. For instance, when using GN, one should be careful with the batch size as the performance of GN degrades with larger batch sizes. Furthermore, Fixup Resnet is limited to Resnets and also requires a carefully-tuned learning rate strategy. Thus, the aforementioned conclusions are more nuanced than previously thought ,hence, more thorough studies and experimentation are needed before concluding if any of these methods can replace BN for different applications and CNN/DNN models.

To address the problem of BN in FL under Non-IID data, we proposed *FedBS*, a new aggregation strategy that handles batch parameters in FL during training to improve the models' performance against heterogeneity. *FedBS* assumes that the global model used in FL has batch normalization layers, and modifies the aggregation strategy at the server side.

In their paper , Y. J. Cho et. al [16] analyze the convergence of *FedAvg* under biased selection strategies and prove that biasing sampling procedure of participating clients towards those with higher loss helps with convergence time as well as higher accuracy. Based on the conclusions drawn from Y. J. Cho et. al analysis, we propose a modification to the naive approach of weighting local models in *FedAvg*. *FedAvg* uses an unbiased selection strategy for clients, where the clients are chosen randomly and their weights are computed based on their local data size. This strategy fails in cases from which we mention when a client has a large data drawn from one class and/or contains outliers. *FedBS* modifies *FedAvg* in two major steps. First, in the early rounds where $F_k(w)$ highly differs from a client to another, *FedBS* weighs the clients according to their respective loss during training. Specifically, the higher the loss of a local model the higher shall be the respective weight of the corresponding client in the aggregation step. This step helps to warm up the global model. When all clients start to converge to the global model and $F_k(w)$ is stable between clients we switch the training using equal weights for all clients and using *FedProx* algorithm. We check the stability of $F_k(w)$ by computing the standard deviation of the loss vector of the partaking clients, if $std(v) \leq \epsilon$ for r consecutive rounds, then we continue the training with *FedProx* algorithm and using equal weight for all clients ($\epsilon = 0.1$ and $r = 5$ for experimentation) (Alg. 1).

Figure 1 provides a global overview for the first step of*FedBS* (without the weight exchange of the global model):

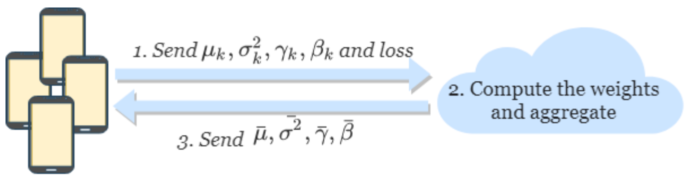
1. Send $\mu_k, \sigma_k^2, \gamma_k, \beta_k$ and loss
2. Compute the weights and aggregate
3. Send $\bar{\mu}, \bar{\sigma}^2, \bar{\gamma}, \bar{\beta}$

Fig. 1. Overview of *FedBS*

1) At each communication round, a subset k of clients is chosen randomly to participate in the current FL round. Each participating client after training on his local data, shares the BN parameters used in the normalization layers with the central server, as well as the loss value obtained in the current round.

2) The server, based on the local loss values, calculates the importance of each client, i.e. higher weight is given to

the client with the highest local loss. Then aggregates all the local models.

3) The updated parameters are sent back to the new selected clients for the next round. Each client updates both training and inference statistics based on those received from the server

Algorithm 1 FEDBS: The K clients are indexed by k, E is the number of local epochs, $\mathcal{B}$ is a set of mini-batches each of size m, η is the learning rate, and ϵ is a positive small number.

```
Server executes:
    initialize $w_0$
    for each round $t = 0, 1, \ldots$ do
        m ← max(C·K, 1)
        $S_t$ ← (select a random set of m clients)
        send $w_t$ to each client k $\in S_t$
        for each client k $\in S_t$ do
            $w_{k,t+1}, F_k(w) \leftarrow ClientUpdate(k, w_t)$
        end for
        $W_k = \frac{F_k(w)}{\sum_{k \in S_t} F_k(w)}$
        $w_{t+1} = \sum_{k \in S_t} W_k w_{t+1}^k$
        until $std(F(w)) < \epsilon$
        switch to FedProx
    end for

ClientUpdate($k, w$):
    for each $i$ from 1 to $E$ do
        for batch $b \in \mathcal{B}$ do
            $w \leftarrow w - \eta \nabla \mathcal{L}(w; b)$
            computer $F_k(w)$
        end for
    end for
    return $w$ and $F_k(w)$ to server
```

V. Experimentation

Datasets and Non-IID settings: We conduct our experiments on three datasets, Mnist, Fashion-Mnist and Cifar-10. For Cifar-10 [27] dataset, We use a CNN model with four convolution layers, a first 5x5 convolution layer followed by three 3x3 convolution layers, with 100, 150, 250, 500 channels respectively. Each convolution layer is followed by a normalization layer, a Relu activation, and a 2x2 max pooling. At the end, three fully connected layers were used (with 4500, 270, 150 units resp.). For Mnist [28] and Fashion-Mnist [29], we use a CNN model with two 5x5 convolution layers, the first one with 16 channels, the second one with 32. Each convolution is followed with BN layer, ReLu activation and a 2x2 max pooling, then a fully connected layer with 1568 units.

We consider two distributions, for Non-IID Data partitions. The first is *balanced label distribution skew* [2] where the Mnist dataset is divided into 200 shards of 300 images. We consider 100 clients, each receives two shards, and thus has samples from only two classes. The same setting is used to partition Fashion-Mnist and Cifar-10. The second is *unbalanced label distribution skew*, we divide the datasets into 1200 shards of 50 samples each. Each client receives at least one shard and a maximum of 30 shards.

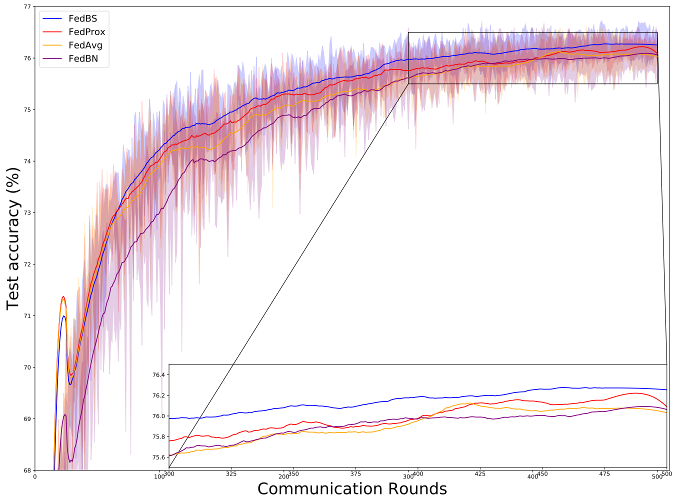

Fig. 2. Testing results on Cifar-10 dataset partitioned unequally.

Baselines We compare the proposed method *FedBS* to two standard algorithms in FL, *FedAvg* and *FedProx*. We also include the recently proposed method *FedBN*. Since *FedBN* does not deal with our setting of testing where the testing clients do not have any data to calculate the mean and variance on, we handle this by taking the average of the mean and the variance learned in the previous round as was done with the trainable parameters in the proposed generalization of *FedBN*.

Training For all datasets, we divide the samples on 100 clients either evenly or unequally. For each communication round, 10 clients are chosen randomly to participate in FL round. During training, we set batch size B to 10, learning rate r to 0.01, local epochs E to 10 and global rounds to 250 for Balanced and unbalanced Mnist/Fashion-Mnist and 1000, 500 for Balanced and Unbalanced Cifar-10 respectively. We use SGD optimizer with cross-entropy loss function. For BN hyperparameters, we use the default values 0.1 for momentum and *ema* as moving average technique.

Evaluation We evaluate *FedBS* on the original test datasets as our general test sets, each contains 10000 samples. For each method, we run the training and test five times and we plot the mean result.

Discussion. Figure 2 and 3 provide the testing accuracy on Cifar-10 dataset in both cases equal and unequal partitioning of the data respectively. We observe that *FedBS* yields up to 2× faster convergence rate with higher test accuracy. We notice that *FedBN* is slower than all other methods including *FedAvg*. Moreover, compared to the baselines, *FedBS* goes smoother with less fluctuations for both cases.

Similarly, we run tests on Mnist and Fashion-Mnist datasets (figure 5 and 6). For both scenarios, equal and unequal partitioning of the data, *FedBS* is on par or better than other FL methods.

Testing on unbalanced Cifar-10 dataset locally (figure 6): We also tested *FedBS* in the local case where clients have access to all BN parameters. Although *FedBN* surpasses the other methods, however, it goes much slower than all of them during the first 200 rounds, which supports the remarks previously made about the slowness of *FedBN*. On the other hand, *FedBS*, *FedAvg*, and *FedProx* all converge to the same accuracy, however, *FedBS* shows faster convergence during earlier rounds.

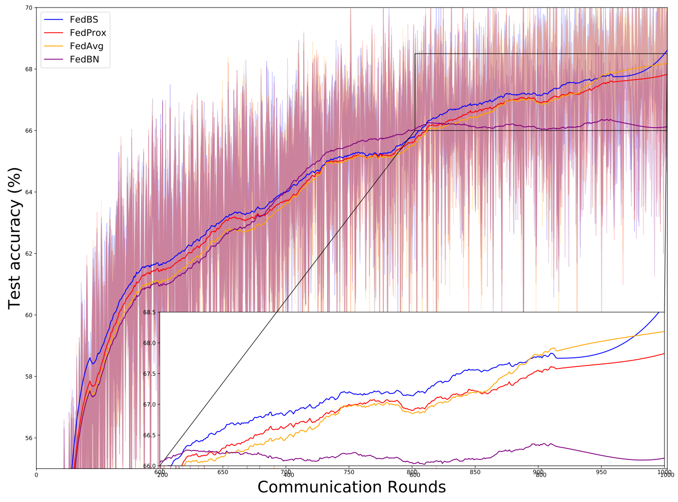

Fig. 3. Testing results on Cifar-10 dataset partitioned equally.

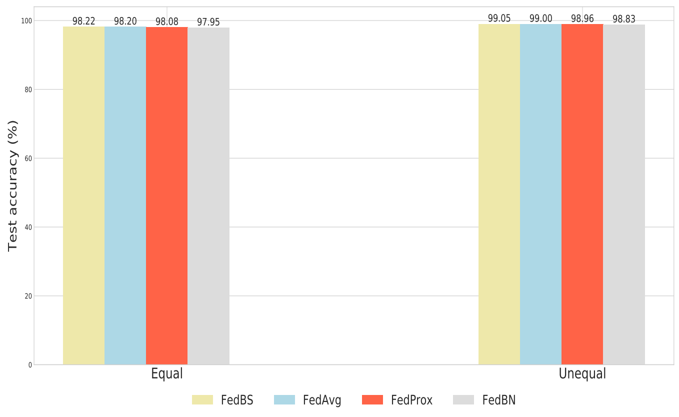

Fig. 4. Testing results on balanced Mnist and Fashion-Mnist datasets.

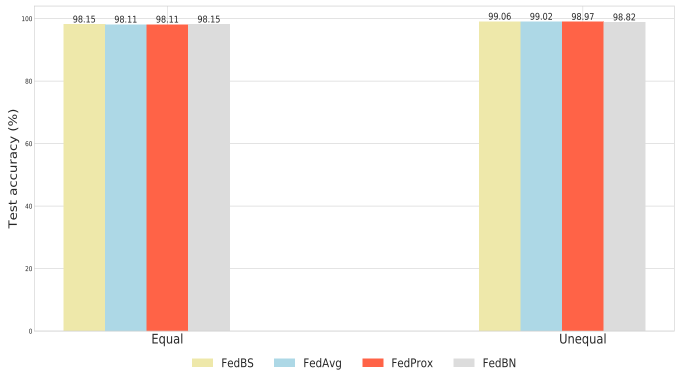

Fig. 5. Experiments results on unbalanced Mnist and Fashion-Mnist datasets.

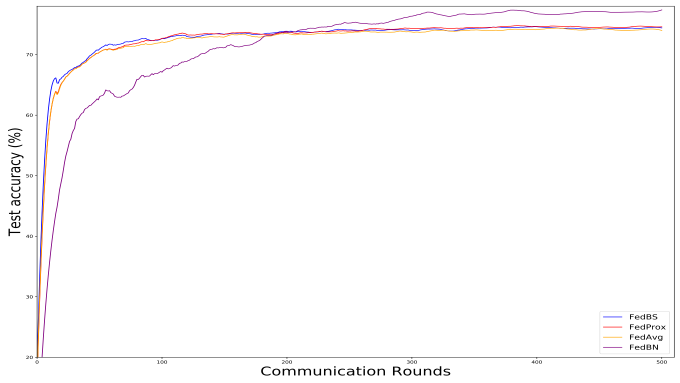

Fig. 6. Testing results on Cifar-10 dataset partitioned unequally (Local test).

VI. Conclusion

Our work *FedBS*, introduces a novel approach for handling DNN models with normalization layers in FL setting. In order to deal with Non-IIDness assumption, *FedBS* warms up the global model by weighting the local models based on each client's loss value, then *FedBS* switches to equal weighting with *FedProx* algorithm. We have comparatively evaluated *FedBS* against *FedAvg*, *FedProx* and *FedBN* using a multitude of datasets under various Non-IID settings.

First, based on exhaustive experimentation conducted on the known and complex dataset Cifar-10, *FedBS* outperformed all the state-of-the-art approaches in term of both convergence rate and model performance with a scale up $2\times$ faster.

Furthermore, throughout all experiments run on Mnist and Fashion-Mnist datasets, *FedBS* was either on par or better than the concurrent methods. Finally, since batch statistics can be used by potential attackers, and as future directions we intend to propose a more secure communication of batch parameters between the clients and the server.

References

[1] "Data volume of internet of things (iot) connections worldwide," https://www.statista.com/statistics/1017863/worldwide-iot-connecteddevices-data-size/, accessed: 2021-02-17.

[2] H. B. McMahan, E. Moore, D. Ramage, and B. A. y Arcas, "Federated learning of deep networks using model averaging," *CoRR*, vol. abs/1602.05629, 2016. [Online]. Available: http://arxiv.org/abs/1602.05629

[3] "Google home," https://store.google.com/product/google_home/, accessed: 2021-02-17.

[4] "Amazon (n.d.) echo & alexa devices. available at:," https://www.amazon.com/Amazon-Echo-And-Alexa-Devices/b?ie=UTF8&node=9818047011/.

[5] Y. Lin, S. Han, H. Mao, Y. Wang, and W. J. Dally, "Deep gradient compression: Reducing the communication bandwidth for distributed training," *CoRR*, vol. abs/1712.01887, 2017. [Online]. Available: http://arxiv.org/abs/1712.01887

[6] J. Konecný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *CoRR*, vol. abs/1610.05492, 2016. [Online]. Available: http://arxiv.org/abs/1610.05492

[7] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for privacy-preserving machine learning," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 1175–1191. [Online]. Available: https://doi.org/10.1145/3133956.3133982

[8] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konecný, S. Mazzocchi, H. B. McMahan, T. V. Overveldt, D. Petrou, D. Ramage, and J. Roselander, "Towards federated learning at scale: System design," *CoRR*, vol. abs/1902.01046, 2019. [Online]. Available: http://arxiv.org/abs/1902.01046

[9] J. Li, M. Khodak, S. Caldas, and A. Talwalkar, "Differentially private meta-learning," *CoRR*, vol. abs/1909.05830, 2019. [Online]. Available: http://arxiv.org/abs/1909.05830

[10] V. Smith, C. Chiang, M. Sanjabi, and A. Talwalkar, "Federated multi-task learning," *CoRR*, vol. abs/1705.10467, 2017. [Online]. Available: http://arxiv.org/abs/1705.10467

[11] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *CoRR*, vol. abs/1502.03167, 2015. [Online]. Available: http://arxiv.org/abs/1502.03167

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

[13] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of fedavg on non-iid data," 2019.

[14] E. Jeong, S. Oh, H. Kim, J. Park, M. Bennis, and S. Kim, "Communication-efficient on-device machine learning: Federated distillation and augmentation under non-iid private data," *CoRR*, vol. abs/1811.11479, 2018. [Online]. Available: http://arxiv.org/abs/1811.11479

[15] A. K. Sahu, T. Li, M. Sanjabi, M. Zaheer, A. Talwalkar, and V. Smith, "On the convergence of federated optimization in heterogeneous networks," *CoRR*, vol. abs/1812.06127, 2018. [Online]. Available: http://arxiv.org/abs/1812.06127

[16] Y. J. Cho, J. Wang, and G. Joshi, "Client selection in federated learning: Convergence analysis and power-of-choice selection strategies," *CoRR*, vol. abs/2010.01243, 2020. [Online]. Available: https://arxiv.org/abs/2010.01243

[17] Y. J. Cho, S. Gupta, G. Joshi, and O. Yagan, "Bandit-based communication-efficient client selection strategies for federated learning," *CoRR*, vol. abs/2012.08009, 2020. [Online]. Available: https://arxiv.org/abs/2012.08009

[18] W. Chen, S. Horvath, and P. Richtárik, "Optimal client sampling for federated learning," *CoRR*, vol. abs/2010.13723, 2020. [Online]. Available: https://arxiv.org/abs/2010.13723

[19] M. Ribero and H. Vikalo, "Communication-efficient federated learning via optimal client sampling," *CoRR*, vol. abs/2007.15197, 2020. [Online]. Available: https://arxiv.org/abs/2007.15197

[20] K. Hsieh, A. Phanishayee, O. Mutlu, and P. B. Gibbons, "The non-iid data quagmire of decentralized machine learning," 2019.

[21] M. Andreux, J. O. du Terrail, C. Beguier, and E. W. Tramel, "Siloed federated learning for multi-centric histopathology datasets," 2020.

[22] X. Li, M. JIANG, X. Zhang, M. Kamp, and Q. Dou, "Fed{bn}: Federated learning on non-{iid} features via local batch normalization," in *International Conference on Learning Representations*, 2021. [Online]. Available: https://openreview.net/forum?id=6YEQUn0QICG

[23] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, R. G. L. D'Oliveira, S. E. Rouayheb, D. Evans, J. Gardner, Z. Garrett, A. Gascón, B. Ghazi, P. B. Gibbons, M. Gruteser, Z. Harchaoui, C. He, L. He, Z. Huo, B. Hutchinson, J. Hsu, M. Jaggi, T. Javidi, G. Joshi, M. Khodak, J. Konecný, A. Korolova, F. Koushanfar, S. Koyejo, T. Lepoint, Y. Liu, P. Mittal, M. Mohri, R. Nock, A. Özgür, R. Pagh, M. Raykova, H. Qi, D. Ramage, R. Raskar, D. Song, W. Song, S. U. Stich, Z. Sun, A. T. Suresh, F. Tramèr, P. Vepakomma, J. Wang, L. Xiong, Z. Xu, Q. Yang, F. X. Yu, H. Yu, and S. Zhao, "Advances and open problems in federated learning," *CoRR*, vol. abs/1912.04977, 2019. [Online]. Available: http://arxiv.org/abs/1912.04977

[24] S. Ioffe, "Batch renormalization: Towards reducing minibatch dependence in batch-normalized models," *CoRR*, vol. abs/1702.03275, 2017. [Online]. Available: http://arxiv.org/abs/1702.03275

[25] Y. Wu and K. He, "Group normalization," *CoRR*, vol. abs/1803.08494, 2018. [Online]. Available: http://arxiv.org/abs/1803.08494

[26] H. Zhang, Y. N. Dauphin, and T. Ma, "Fixup initialization: Residual learning without normalization," *CoRR*, vol. abs/1901.09321, 2019. [Online]. Available: http://arxiv.org/abs/1901.09321

[27] A. Krizhevsky, "Learning multiple layers of features from tiny images," *University of Toronto*, 05 2012.

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

[29] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *CoRR*, vol. abs/1708.07747, 2017. [Online]. Available: http://arxiv.org/abs/1708.07747