Naas, Si Ahmed; Sigg, Stephan

Fast converging Federated Learning with Non-IID Data

Published in:
2023 IEEE 97th Vehicular Technology Conference, VTC 2023-Spring - Proceedings

DOI:
10.1109/VTC2023-Spring57618.2023.10200108

Published: 01/01/2023

Document Version
Peer-reviewed accepted author manuscript, also known as Final accepted manuscript or Post-print

Please cite the original version:
Naas, S. A., & Sigg, S. (2023). Fast converging Federated Learning with Non-IID Data. In *2023 IEEE 97th Vehicular Technology Conference, VTC 2023-Spring - Proceedings* (IEEE Vehicular Technology Conference; Vol. 2023-June). IEEE. https://doi.org/10.1109/VTC2023-Spring57618.2023.10200108

Fast converging Federated Learning with Non-IID Data

Si-Ahmed Naas
Dept. Communication and Networking
Aalto University
Espoo, Finland
si-ahmed.naas@aalto.fi

Stephan Sigg
Dept. Communication and Networking
Aalto University
Espoo, Finland
stephan.sigg@aalto.fi

***Abstract*—With the advancement of device capabilities, Internet of Things (IoT) devices can employ built-in hardware to perform machine learning (ML) tasks, extending their horizons in many promising directions. In traditional ML, data are sent to a server for training. However, this approach raises user privacy concerns. On the other hand, transferring user data to a cloud-centric environment results in increased latency. A decentralized ML technique, Federated learning (FL), has been proposed to enable devices to train locally on personal data and then send the data to a server for model aggregation. In these models, malicious devices, or devices with a minor contribution to a global model, increase communication rounds and resource usage. Likewise, heterogeneous data, such as non-independent and identically distributed (Non-IID), may decrease accuracy of the FL model. This paper proposes a mechanism to quantify device contributions based on weight divergence. We propose an outlier-removal approach which identifies irrelevant device updates. Client selection probabilities are computed using a Bayesian model. To obtain a global model, we employ a novel merging algorithm utilizing weight shifting values to ensure convergence towards more accurate predictions. A simulation using the MNIST dataset employing both non-iid and iid devices, distributed on 10 Jetson Nano devices, shows that our approach converges faster, significantly reduces communication cost, and improves accuracy.**

***Index Terms*—Federated learning, communication reduction, non-iid data, edge computing, fog networks, Internet of things.**

I. Introduction

With the explosive growth in IoT devices, massive amounts of data are being generated. In 2025, more than 25 billion devices will be connected, and this number is expected to rise to 80 billion by 2030 [1]. By 2025, 180 trillion gigabytes of data are expected to be generated [2]. In the traditional cloud-centric approach, all devices send their data to a centralized node for training. However, this approach poses several challenges regarding resource requirements and user privacy [3]. Federated Learning (FL) has gained significant attention recently. It solves these challenges by allowing devices to train their data locally. In each round, devices send local updates to a server which then aggregates the received weights. FL allows learning from distributed data without exposing user data and is currently being used in health care, transportation [4], and autonomous driving [5]. Although FL can receive input from different users, training is affected by the overall data distribution [6]. It was shown in [7] that the accuracy of FedAvg [8] is significantly reduced on the non-independent-and-identically-distributed (non-iid) CIFAR-10 dataset [9]. A major FL challenge is data heterogeneity among the clients [10]. Non-iid data slows model training [11]. The model may even diverge if deployed over non-iid samples [12] (cf. Fig 1). In realistic settings, devices acquire different data and classes [6] due to various types of sensor equipment.

Several authors have proposed solutions to the non-iid data issues, such as [7], [11], [13]–[21]. In [13], the authors propose a modification to the FedAvg algorithm [8] designed to improve the model convergence by adding a proximal term to the objective function. This approach does not consider the data imbalance in client selections. SCAFFOLD [14] adds a drift in the local training to correct local updates. However, the number of communication rounds needed for this approach is significantly greater than that for the original scheme. The authors of [7] measured the weight divergence using an earth mover's distance metric of both global and local data distributions. The authors propose to share a subset of data with all nodes to reduce divergence. However, this approach requires that the size of transmitted data be defined and the communication load be increased; furthermore, sharing sensitive data with all nodes violates user privacy. In [15], the authors propose to select a subset of users to create iid data. This approach requires that some users upload their data to the server for model aggregation. However, uploading data from a device may also expose sensitive data. Similarly, in [16] a data augmentation technique is proposed to reduce the global data imbalance. The authors employ mediators to reschedule client training based on KullbackLeibler Divergence [22]. However, assuming that users will share their data is not practical. The authors of [17] address FL training over a wireless network. They focus on selecting the most relevant subset of users for training to minimize the loss function. In addition, the authors study the relationship between packet error rates and FL performance. However, they focus on the communication aspects rather than the data distribution aspects. In [18], a reputation-based client selection mechanism is based on a multi-weight subjective logic model. The authors also propose a blockchain scheme for managing clients' reputations in a decentralized manner. They employ a contract theory approach to frequently

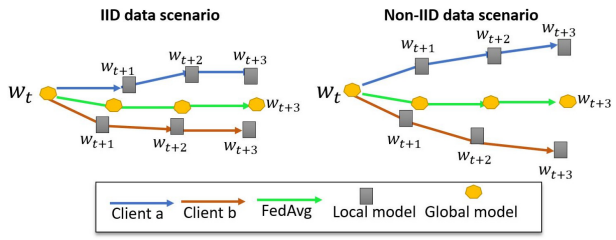


Fig. 1. Comparison between update drifts in iid and non-iid settings

select devices that have high accuracy to participate in FL. In addition, a Semi-Asynchronous Federated Averaging (SAFA) protocol is proposed in [11] for efficient client selection based on their impact on overall learning. The approach also adopts a lag-tolerant mechanism to address the trade-off between faster convergence and communication rounds. In [19], a convergence rate comparison is conducted between random scheduling (RS), round-robin (RR), and proportional fair (PF) scheduling schemes. The work reveals that under a fixed spectrum, a tradeoff between the number of devices and the subchannel bandwidth exists.

Most work has focused on the communication aspect or IoT device characteristics rather than data distribution to solve the non-iid issue, such as in [16]–[18], or redistributing data to other users, which exposes sensitive data, such as in [7]. In [7], the global set is uploaded to all devices with high weight divergence, which increases communication rounds and decreases user privacy. A few authors have considered data heterogeneity. In [6], weight divergence is used to determine non-iid clients. However, client selection relies on a to-be-determined selection factor.

This paper proposes a mechanism that addresses data heterogeneity issue, improving classification accuracy and accelerating model convergence. The main contributions of our work are as follows:

- We employ weight divergence to measure node contributions to the global model.
- We propose client selection based on a Bayesian model.
- To discard nodes with lower contribution, we employ an outlier removal mechanism that does not require a predefined threshold.
- We propose a merging mechanism based on adapting model weights.

Our scheme measures the weight divergence between each device and the global model, before the weight divergence values are fed to the proposed outlier mechanism. After outlier identification, the device selection probability is updated using the beta function. Devices with regular positive updates will have higher selection probabilities, while those with negative updates will be assigned lower probabilities (cf. Fig 2).

II. System Model

Let $\mathcal{F}$ be a set of devices. Users are denoted by $u \in \mathcal{U}$ and the training dataset is $\mathfrak{D} = \{x, y\}$. Each device possesses a

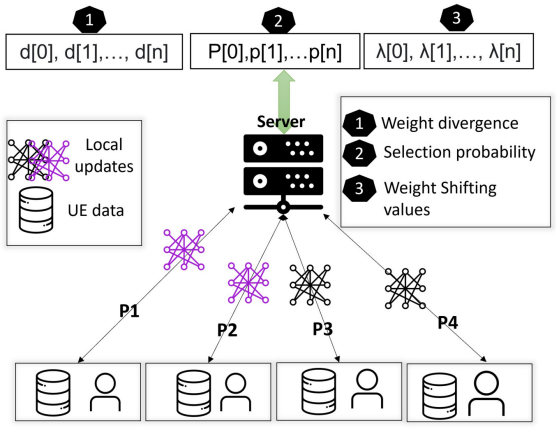


Fig. 2. Overview of proposed FL scheme. It consist of three phases,: (a) measuring weight divergence, (b) calculating selection probability, and (c) finding weight shifting values with which to merge the global model.

portion of data $\mathfrak{D}_k$, and transmit models to base station, where $\mathfrak{D}_k = \{(x_k, y_k)\}$, $k \in \mathbb{N}$. Each device k in round i locally updates the model as follows:

$$w_i^k = w_{i-1}^k - \eta \nabla \mathrm{L}\left(w_{i-1}^k\right) \tag{1}$$

where L is the loss function, and η is the learning rate.

The communication among IoT devices takes place through wireless access points. The bandwidth allocated to packets transmission is B^i for each communication round. Each packet occupies 60 KHz and 0.25 ms according to 3GPP 5G specification. We consider a log-distance path-loss model with log-normal shadowing [1], namely $PL_{[dB]} = 140.7 + 36.7 \log_{10} d_{[km]} + \mathcal{N}(8)$ [23]. The transmission rate of the access points $a_i \in \mathcal{A}$ is

$$r_i = \eta_i B^i \log_2 \left(1 + \frac{p_i^{tx} - PL_{[dB]}}{\sigma^2}\right) \tag{2}$$

where $\eta_i \in [0, 1]$ denotes the fraction of the allocated bandwidth to the access point for data sending, p_i^{tx} is the transmission signal power, σ^2 is the noise power.

The communication between devices and base stations follows Eq. 2. The energy (ϑ) of IoT device required to offload a data d at a round i is

$$\vartheta_i = p_i^{tx} \delta_i^d \tag{3}$$

A. Non-iid and weight divergence

Non-iid data can affect the FedAvg algorithm [8] performance due to the disparities among the local and the global distributions. In addition, weight averaging at the server often suffers when the data in the devices is heterogeneous [24].

The weight divergence among iid data should be smaller than that in non-IID data [6]. Therefore, the rate of divergence can be employed to distinguish non-iid clients (Alg. 1). The weight divergence is defined as follows:

$$div_i^k \leftarrow \left\| w_i^k - \widetilde{w_i} \right\| \tag{4}$$

Algorithm 1: Learning based client selection

Input : A set of local updates $d \in \mathcal{D}$, λ
Output : group_A, group_B

group_A $\leftarrow \varnothing$
group_B $\leftarrow \varnothing$
// Local device update
foreach $d \in \mathcal{D}$ **do**
  foreach $epoch \in \mathcal{E}$ **do**
    $w_{i+1} \leftarrow (d_i, w_i)$
    $w_{i+1} \leftarrow w_i - \eta \nabla \mathrm{L}(w_i)$
// For each communication round
foreach $c \in \mathcal{C}$ **do**
  foreach $d \in \mathcal{D}$ **do**
    $\mathrm{div}_i^k \leftarrow \left\| w_i^k - \widetilde{w_i} \right\|$
    IF div_i^k is an outlier
      group_A $\leftarrow add(d)$
    ELSE
      $\underline{group}$_B $\leftarrow add(d)$
return group_A, group_B

Algorithm 2: Weight divergence outlier detection algorithm

Input : A set of weight divergence values $d \in \mathcal{D}$, $\lambda, Y, Shift_{vec}$
Output : Outlier score ∂

// Compute KNN
foreach $\mathcal{X} \in div_i$ **do**
  $S_k \leftarrow$ ComputeNearestNeighbors(X,div_i)
  // Calculates the mean of neighbors
  $\tilde{m} = \frac{\sum_{i=0}^{N} S_k^i}{\left\| \vec{S_k} \right\|}$
  // Replace by the mean of the neighbors
  $\mathcal{X}' \leftarrow \tilde{m}$
  $\underline{\mathrm{Y}} \leftarrow add(\mathrm{X}')$
foreach $\mathcal{X}, \mathcal{X}' \in div_i, Y$ **do**
  Shift$_{vec} \leftarrow |\mathcal{X} - \mathcal{X}'|$
return Shift$_{vec}$, $std_v(Shift_{vec})$

where w_i^k is the weight of client k at round i, and $\widetilde{w_i}$ is the weight for global model. When the weight divergence is large, the model performance will be lower [25], [26]. The time complexity for the Alg. 1 is $\mathcal{O}(n)$.

Employing the bounding techniques found in [25], [27], the weight divergence at round i for clients k and k' can be expressed as:

$$\left\| w_i^{k'} - w_i^k \right\| \leq \eta g_{max}(\widetilde{w_0}) \sum_{j=1}^{C} \left\| p^{k'}(y=j) - p^k \right\| \quad (5)$$

where $g_{max}(\widetilde{w}) = max_{j=1}^{C} \left\| \nabla_{\widetilde{w}} \mathbb{E}_{x|y=j} \left[\log f_j(x, \widetilde{w}) \right] \right\|$, C represents the classes, p^k is the data distribution of set samples (x,y) on device k, and $\widetilde{w}$ is the initial global model weights. The previous equation 5 shows the relationship between weight divergence and the data distribution; thus, we employ weight divergence in client selection.

B. Client selection

Using a random client selection mechanism during FL learning can result in the inclusion of clients without significant improvements, thus increasing the communication overhead. We propose to employ the beta distribution function of a variable x in this work which is defined as:

$$f(x) = \begin{cases} \frac{1}{B(a,b)} x^{a-1}(1-x)^{b-1} & \text{IF } 0 < x < 1 \\ 0 & \text{ELSE} \end{cases} . \quad (6)$$

where $B(a,b) = \int_0^1 x^{a-1}(1-x)^{b-1} dx$

The mean of the beta distribution of a variable X is defined as:

$$E[X] = \frac{\alpha}{\alpha + \beta} \quad (7)$$

where $\alpha = \alpha_0 + x_\alpha$, $\beta = \beta_0 + x_\beta$; x_α is the number of positive updates by a client k, and x_β represents the negative updates, α_0 and β_0 are expected prior probabilities in which we consider $\alpha_0 = \beta_0$.

When Alg.2 determines that if the weight divergence of an update is an outlier, it is considered to be a negative update. An outlier is identified if a standard deviation of a point and its shifted value is higher than the overall standard deviation $|\mathcal{X} - \mathcal{X}'| > std_v$ (which is computed from all shifted values). The std_v is considered as a a threshold outlier value. To avoid excluding high-performing updates, which may be identified as outliers, the points showing significant contributions to the global model are removed from the outlier list. The parameters of the beta function are defined as follows:

$$x_\alpha, x_\beta = \begin{cases} x_\alpha = x_\alpha + 1 & \text{IF } |\mathcal{X} - \mathcal{X}'| < std_v \\ x_\beta = x_\beta + 1 & \text{ELSE} \end{cases} \quad (8)$$

where $\mathcal{X}, \mathcal{X}'$ are the original weight divergence value and its shifted value, std_v is the standard deviation of all shifted values according to Alg.2, respectively.

C. Blocking malicious devices

Malicious devices or devices with meager updates negatively affect FL. Therefore, a weight divergence metric is used to quantify contributions to the overall learning, and an outlier detection mechanism is adopted from [28]. Any device update that is considered to be an outlier i.e., a negative update, will be further inserted into the beta distribution function to update the selection probability. After frequent negative updates, the selection probability is significantly reduced, excluding non-useful devices.

D. Adaptive weight shifting at the server

In addition to FedAvg [8], we propose a new technique for weight merging at the server. Our proposed merging algorithm is described in Alg.3. After the local updates are sent to the server, the combination of the received weights must still be determined for the next communication round. Let $\widetilde{w}_i$ be the global model at the server at round i, and w_i^k the weights acquired from client k at round i, $\forall k$ such that $0 \leq k < n$. The new weights $\widetilde{w}_{i+1}$ are then calculated as follows (Alg.3):

Algorithm 3: Adaptive-weight shifting federated learning algorithm

Input : A set of local updates $d \in \mathcal{D}$, λ_k: weight shifting value, $\widetilde{W}$: global model weight.
Output : Global model FL_{acc} loss

selection_proba $\leftarrow \varnothing$;
selection_proba $\leftarrow \bar{c}ompute\_proba()$;
D=select_devices(selection_proba);
// Local device update
foreach $d \in \mathcal{D}$ **do**
  foreach $epoch \in \mathcal{E}$ **do**
    $w_{i+1} \leftarrow (d_i, w_i)$;
    $w_{i+1} \leftarrow w_i - \eta \nabla \mathrm{L}(w_i)$
// For each communication round
foreach $i \in \mathcal{C}$ **do**
  foreach $k \in \mathcal{D}$ **do**
    $\lambda_k \leftarrow 1.0$
    $\tau \leftarrow 0.2$
    vals $\leftarrow []$
    while $\lambda_k > 0$
      $\lambda_k, \widetilde{W} \leftarrow \left(w_i^k - \widetilde{w}_i\right)$
      vals $\leftarrow add(\lambda_k)$
      vals $\leftarrow evaluate(\widetilde{W})$
      $\underline{\lambda}_k \leftarrow \lambda_i^d - \tau$
    $\underline{\lambda}_k \leftarrow \max(vals)$
  $\widetilde{w}_{i+1} = \widetilde{w}_i + \frac{1}{n} \sum_{k=1}^{n} \lambda_k \left(w_i^k - \widetilde{w}_i\right)$
  return $\widetilde{w}_{i+1}$

$$\widetilde{w}_{i+1} = \widetilde{w}_i + \frac{1}{n} \sum_{k=1}^{n} \lambda_k \left(w_i^k - \widetilde{w}_i\right) \tag{9}$$

Where λ_k are tradeoff parameters such that $0 \leq \lambda_k \leq 1$, $\forall k \in \{1, \ldots, n\}$. The λ_k parameter characterize the impact of the new weights with two special cases to consider, namely, $\lambda_k = 0$, which completely ignores the weights from client k ; and $\lambda_k = 1$, which entirely replaces the global model weight with that of client k.

III. Simulation results

In this section, we describe our evaluation settings. We prepare ten Jetson Nanos, small devices with 128 NVIDIA CUDA cores, a Quad-core ARM Cortex-A57 MPCore processor, 4 GB 64-bit LPDDR4 of RAM, and 16 GB of storage (the testbed is shown in Fig. 3a). The Jetson Nano devices support the popular machine learning framework and libraries such as TensorFlow, OpenCV and Keras. We employ a laptop as a server with 32 GB of RAM and i7-7700HQ CPU. To measure the performance of our approach, we compute the classification accuracies of the obtained models. In our experiment, we divide the devices into iid devices and non-iid devices. Let K be the number of devices, and ξ be the fraction of non-iid devices (as shown in Fig. 3b).

We propose two evaluation scenarios, one a hybrid with iid and non-iid devices, and the other containing just iid devices.

TABLE I
Number of communication rounds needed to achieve T_A (target accuracy) for $\xi = 0.3$

Scheme/target accuracy	87%	88%	88.5%	89.2%	89.4%	89.8%
Original FL scheme	5	9	14	29	NaN	NaN
Outlier removal FedAvg scheme	3	5	6	15	20	36

TABLE II
Number of communication rounds needed to achieve T_A (target accuracy) for $\xi = 0.4$

Scheme/target accuracy	85%	86%	87.5%	88.5%	89.5%	89.7%
Original FL scheme	4	5	8	16	-	-
Outlier removal FedAvg scheme	2	3	5	20	28	37

TABLE III
Number of communication rounds needed to achieve T_A (target accuracy) for $\xi = 0.5$

Scheme/target accuracy	85%	86%	87.5%	88.4%	88.5%	88.8%
Original FL scheme	8	10	27	-	-	-
Outlier removal FedAvg scheme	2	2	5	12	14	24

TABLE IV
Number of communication rounds needed to achieve T_A (target accuracy) for $\xi = 0.7$

Scheme/target accuracy	85%	85.5%	86%	86.8%	87%	87.6%
Original FL scheme	22	29	44	-	-	-
Outlier removal FedAvg scheme	2	2	3	4	5	13

TABLE V
Number of communication rounds needed to achieve T_A (target accuracy) for $\xi = 0.8$

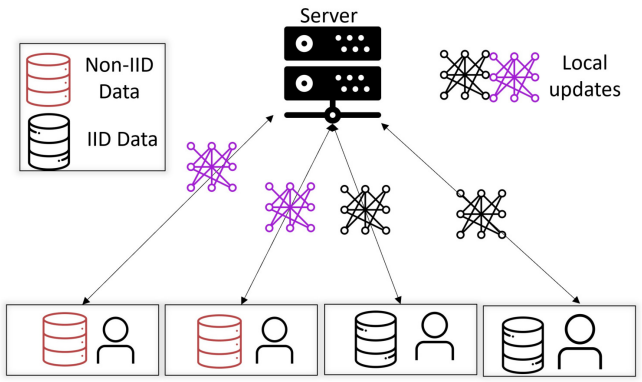
Scheme/target accuracy	82.5%	83.5%	84.5%	85.8%	86%	86.2%
Original FL scheme	32	-	-	-	-	-
Outlier removal FedAvg scheme	1	2	3	7	8	13

A. Comparative scheme

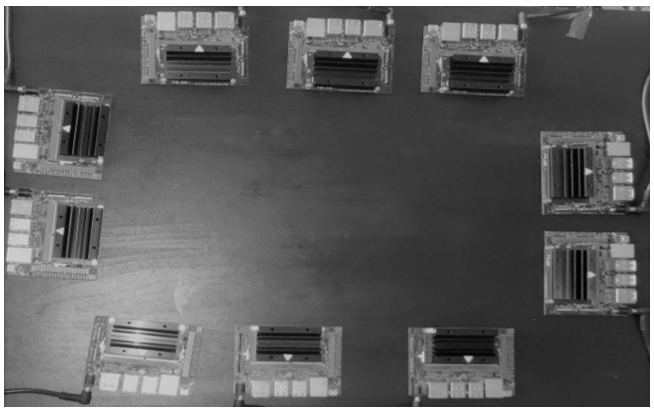
We compare our outlier removal scheme with the traditional FedAvg algorithm. We also compare our averaging technique (adaptive weight shifting in Alg.3) with FedAvg algorithm [8]. Finally, we evaluate the communication and the energy cost of our proposed scheme.

B. Model Architecture

We use a CNN for MNIST dataset [29] classification and testing, a widely used dataset, with 60,000 training samples. The model is implemented using Keras and TensorFlow. It consists of two one-Dimensional convolutional layers. The first layer has 200 neurons and the second convolutional layer is trained with 100 neurons. Both layers trained with a ReLU activation function. The last dense layer is trained with a Softmax activation function. Adam optimizer is employed at a learning rate of 0.001 and ϵ set to 1e-07. We design scenarios where devices hold both iid and non-iid data and other scenarios where only iid devices exist.

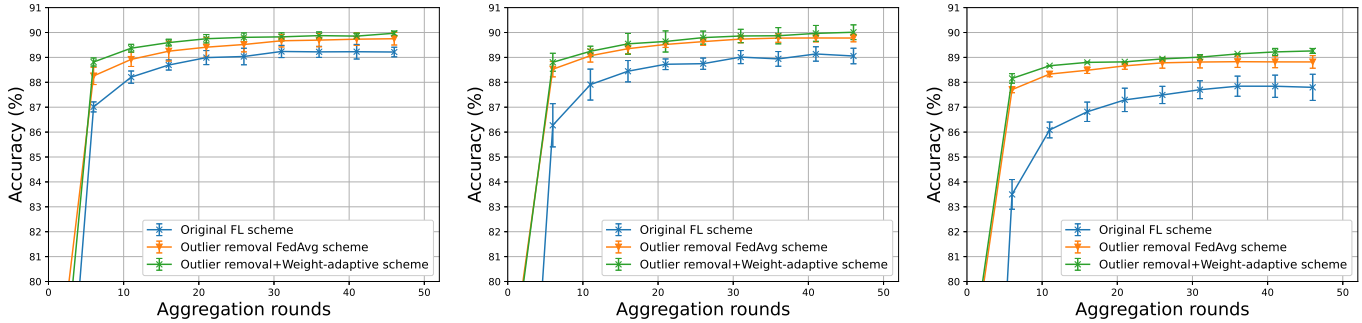


(a) Illustration of system learning; the server receives various weights for the clients, and then detect outlier updates before constructing the global model.



(b) Environmental testbed setting with 10 Jetson Nano IoT devices

Fig. 3. Illustration of the experimental setting and distribution strategy utilized in our case study. 10 Jetson Nano IoT devices are utilized as remote devices storing the data while a server is computing the model aggregations.



(a) 30% of devices are non-iid

(b) 40% of devices are non-iid

(c) 50% of devices are non-iid

Fig. 4. The impact of device outliers removal on the classification accuracy; note that the y-axis range in figures (a), (b), (c) is adapted to the respective gains

C. Scenario1: training with hybrid devices

To evaluate the case of hybrid iid and non-iid devices, we consider the following proportions of non-iid devices, $\xi \in \{0.3, 0.4, 0.5, 0.7, 0.8\}$.

1) Classification accuracy: Fig. 4 and Fig. 5 depict the comparison of the proposed approach to the traditional FL scheme. We also compare our weight adaptive mechanism to the FedAvg algorithm. The results demonstrate an increase in accuracy compared to traditional FL. Our approach significantly reduces the communication rounds and converges faster (Table I-Table V). In addition, our adaptive weight approach improves the classification accuracy further.

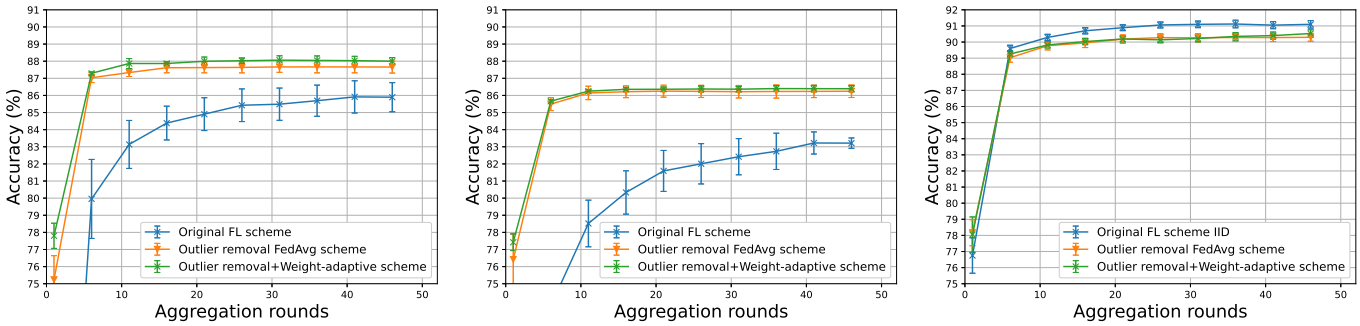
2) Communication and energy cost: We measure the computational and communication for both our scheme and traditional FL scheme. For the case of $\xi = 0.7$, our scheme required 2408 Kb compared to 3440 Kb for the traditional FL. In addition, our scheme converged in 13 communication rounds compared to 44 rounds for traditional FL. Both schemes have average energy consumption of ϑ =2.3 watts (recorded every two seconds) which is adequate for IoT scenarios.

D. Scenario2: training with iid devices

We consider a set of devices constituting only iid data in this scenario. The experiment depicted in Fig.5c demonstrates that when our approach was applied to devices with iid data only, its performance remained very close to that of traditional FL.

IV. Conclusion

We have proposed a scheme to improve FL on non-iid data. We considered weight divergence as a measure of a node's contribution. To recognize negative updates from weight divergences values, we proposed an outlier detection mechanism that does not require a threshold. We employ a Bayesian model to compute nodes' selection probability. We have also proposed a weight merging approach for the server, thus improving accuracy. The obtained results show that our system can improve the classification accuracy and significantly accelerate the FL convergence for non-iid scenarios. Furthermore, our system decreased the communication rounds required for convergence, thus reducing FL latency and computation resource requirements. Our approach is particularly promising for applications that collect data generated from different



(a) 70% of devices are non-iid (b) 80% of devices are non-iid (c) IID distribution

Fig. 5. The impact of device outliers removal on the classification accuracy; note that the y-axis range in figures (a), (b), (c) is adapted to the respective gains

sensors (such as in healthcare scenarios), or applications that perform online data collection.

REFERENCES

[1] Si-Ahmed Naas, Thaha Mohammed, and Stephan Sigg. A global brain fuelled by local intelligence: Optimizing mobile services and networks with ai. In *2020 16th International Conference on Mobility, Sensing and Networking (MSN)*, pages 23–32, 2020.

[2] Canh T. Dinh, Nguyen H. Tran, Minh N. H. Nguyen, Choong Seon Hong, Wei Bao, Albert Y. Zomaya, and Vincent Gramoli. Federated learning over wireless networks: Convergence analysis and resource allocation. *IEEE/ACM Transactions on Networking*, 29(1):398–409, 2021.

[3] Shashi Raj Pandey, Nguyen H. Tran, Mehdi Bennis, Yan Kyaw Tun, Aunas Manzoor, and Choong Seon Hong. A crowdsourcing framework for on-device federated learning. *IEEE Transactions on Wireless Communications*, 19(5):3241–3256, 2020.

[4] Jianxin Zhao, Xinyu Chang, Yanhao Feng, Chi Harold Liu, and Ningbo Liu. Participant selection for federated learning with heterogeneous data in intelligent transport system. *IEEE Transactions on Intelligent Transportation Systems*, pages 1–10, 2022.

[5] Hongyi Zhang, Jan Bosch, and Helena Holmström Olsson. End-to-end federated learning for autonomous driving vehicles. In *International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2021.

[6] Wenyu Zhang, Xiumin Wang, Pan Zhou, Weiwei Wu, and Xinglin Zhang. Client selection for federated learning with non-iid data in mobile edge computing. *IEEE Access*, 9:24462–24474, 2021.

[7] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. Federated learning with non-iid data. *CoRR*, abs/1806.00582, 2018.

[8] H. Brendan McMahan, Eider Moore, Daniel Ramage, and Blaise Agüera y Arcas. Federated learning of deep networks using model averaging. *CoRR*, abs/1602.05629, 2016.

[9] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

[10] Xiaoxiao Li, Meirui Jiang, Xiaofei Zhang, Michael Kamp, and Qi Dou. Fedbn: Federated learning on non-iid features via local batch normalization. *CoRR*, abs/2102.07623, 2021.

[11] Wentai Wu, Ligang He, Weiwei Lin, Rui Mao, Carsten Maple, and Stephen Jarvis. Safa: A semi-asynchronous protocol for fast federated learning with low overhead. *IEEE Transactions on Computers*, 70(5):655–668, 2021.

[12] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank J. Reddi, Sebastian U. Stich, and Ananda Theertha Suresh. SCAFFOLD: stochastic controlled averaging for on-device federated learning. *CoRR*, abs/1910.06378, 2019.

[13] Anit Kumar Sahu, Tian Li, Maziar Sanjabi, Manzil Zaheer, Ameet Talwalkar, and Virginia Smith. On the convergence of federated optimization in heterogeneous networks. *CoRR*, abs/1812.06127, 2018.

[14] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank J. Reddi, Sebastian U. Stich, and Ananda Theertha Suresh. SCAFFOLD: stochastic controlled averaging for on-device federated learning. *CoRR*, 2019.

[15] Naoya Yoshida, Takayuki Nishio, Masahiro Morikura, Koji Yamamoto, and Ryo Yonetani. Hybrid-fl for wireless networks: Cooperative learning mechanism using non-iid data. In *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, pages 1–7, 2020.

[16] Moming Duan, Duo Liu, Xianzhang Chen, Yujuan Tan, Jinting Ren, Lei Qiao, and Liang Liang. Astraea: Self-balancing federated learning for improving classification accuracy of mobile deep learning applications. In *2019 IEEE 37th International Conference on Computer Design (ICCD)*, pages 246–254, 2019.

[17] Mingzhe Chen, Zhaohui Yang, Walid Saad, Changchuan Yin, H. Vincent Poor, and Shuguang Cui. A joint learning and communications framework for federated learning over wireless networks. *IEEE Transactions on Wireless Communications*, 20(1):269–283, 2021.

[18] Jiawen Kang, Zehui Xiong, Dusit Niyato, Shengli Xie, and Junshan Zhang. Incentive mechanism for reliable federated learning: A joint optimization approach to combining reputation and contract theory. *IEEE Internet of Things Journal*, 6(6):10700–10714, 2019.

[19] Howard H. Yang, Zuozhu Liu, Tony Q. S. Quek, and H. Vincent Poor. Scheduling policies for federated learning in wireless networks. *IEEE Transactions on Communications*, 68(1):317–333, 2020.

[20] Tian Li, Maziar Sanjabi, and Virginia Smith. Fair resource allocation in federated learning. *CoRR*, abs/1905.10497, 2019.

[21] Anit Kumar Sahu, Tian Li, Maziar Sanjabi, Manzil Zaheer, Ameet Talwalkar, and Virginia Smith. On the convergence of federated optimization in heterogeneous networks. *CoRR*, abs/1812.06127, 2018.

[22] Shuyi Ji, Zizhao Zhang, Shihui Ying, Liejun Wang, Xibin Zhao, and Yue Gao. Kullback-leibler divergence metric learning. *IEEE Transactions on Cybernetics*, pages 1–12, 2020.

[23] Antti Toskala. 5g standards and outlook for 5g unlicensed. https://www.multefire.org/wp-content/uploads/5G_Standard_Toskala_MUulteFire-Open-Day-Meeting.pdf, 6 2018.

[24] Xiaoxiao Li, Meirui Jiang, Xiaofei Zhang, Michael Kamp, and Qi Dou. Fedbn: Federated learning on non-iid features via local batch normalization, 2021.

[25] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. Federated learning with non-iid data. *CoRR*, abs/1806.00582, 2018.

[26] Jinjin Xu, Wenli Du, Ran Cheng, Wangli He, and Yaochu Jin. Ternary compression for communication-efficient federated learning. *CoRR*, abs/2003.03564, 2020.

[27] Hao Wang, Zakhary Kaplan, Di Niu, and Baochun Li. Optimizing federated learning on non-iid data with reinforcement learning. In *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*, pages 1698–1707, 2020.

[28] Jiawei Yang, Susanto Rahardja, and Pasi Fränti. Mean-shift outlier detection and filtering. *Pattern Recognition*, 115:107874, 2021.

[29] Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.