Layer-Wise Adaptive Weighting for Faster Convergence in Federated Learning

Vedant S. Lanjewar
School of Science, Engineering and Technology
Penn State Harrisburg, The Pennsylvania State University
Middletown, PA 17057, United States
vsl5052@psu.edu

Hai-Anh Tran
School of Information and Communications Technology
Hanoi University of Science and Technology (HUST)
Hanoi, Vietnam
anhth@soict.hust.edu.vn

Truong X. Tran
School of Science, Engineering and Technology
Penn State Harrisburg, The Pennsylvania State University
Middletown, PA 17057, United States
truong.tran@psu.edu

***Abstract*—Federated learning (FL), a decentralized approach utilizing multiple devices for training machine learning algorithms, presents an innovative solution to the challenges associated with handling sensitive personal data in neural network training. This research explores the application of FL for training neural network models, emphasizing its benefits in safeguarding privacy, optimizing efficiency, and addressing communication costs. While FL offers advantages such as enhanced data efficiency, promotion of heterogeneity, and scalability, challenges arise from the presence of non-independent and non-identically distributed data in the real world. This research introduces the FedLayerWise algorithm, a novel approach that dynamically assigns weights to each layer of clients' models based on their contributions to the global model. The algorithm leverages the relationship between gradients of the client model and the global model to adaptively modify weights post-training, expediting convergence towards optimal loss. Experimental results on the MNIST dataset with non-identical distributions demonstrate the algorithm's effectiveness. FedLayerWise outperforms existing algorithms (FedAdp and FedAvg) under highly skewed non-IID data distribution, achieving faster convergence. The proposed modifications showcase improvements and converge faster to non-IID data distributions, reducing computation rounds by about 40% compared to FedAdp and about 50% compared to FedAvg in such scenarios.**

***Index Terms*—Federated Learning, Neural Networks, Fast Convergence Rate, Machine Learning Algorithm**

I. Introduction

Federated learning (FL) is a machine learning approach that takes advantage of multiple devices to train machine learning models, such as artificial neural networks (ANN). It employs a decentralized approach, where instead of moving data to the local agent location, the model is brought to the data. This research will primarily delve into the application of FL for training ANN models. FL addresses the challenge of handling sensitive personal data required for training ANNs. In traditional centralized approaches, rich data necessary for ANN training often contains personal and sensitive information, making it problematic to transfer to a centralized data center. However, FL offers an innovative solution to this issue. In FL, the ANN model is deployed and remains on individual devices, allowing training to occur using the data present on these devices. The key advantage is that only the model weights are transmitted during the training process rather than the raw data itself. This architectural shift significantly reduces communication costs, typically achieving reductions by a factor of 10 to 100 times when compared to traditional synchronized stochastic gradient descent [1]. By decentralizing the model and focusing on sending only model weights, FL not only safeguards personal and sensitive data but also optimizes the efficiency of the training process, making it a promising approach in the fields of software, artificial intelligence, machine learning, and data science. Nevertheless, one of the primary challenges in FL revolves around communication costs associated with transferring weights between client and global models. Striking a balance between the number of connected clients and communication costs is crucial. An increase in the number of clients connected to the central server inevitably leads to a proportional increase in communication costs.

Recent research has shed light on the fact that certain clients provide more valuable updates than others in reducing the overall model loss [2]. Furthermore, there is a correlation between the data distribution on a node and its contribution to the aggregation of the global model, which can be harnessed by utilizing gradient information as in FedAdp [3]. However, FedAdp aims to utilize the angle between the gradient of the local ANN model and the gradient of the global model, which may not reflect the diversity of each layer in the ANN model.

In this research, we aim to expedite the convergence of the training algorithm towards optimal loss. Our approach involves leveraging the relationship between the gradients at each layer of the client model and the global model. Through this, we will adaptively modify the weight of the client model post-training and aggregate these modified weights to obtain the global model.

The contribution of our work is as follows:

- Introduces FedLayerWise Algorithm: This research proposes a novel algorithm called FedLayerWise that dynam-

2835-5776/24/$31.00 ©2024 IEEE
DOI 10.1109/IRI62200.2024.00036

ically assigns weights to different layers of client models during training. This approach helps the model converge faster towards optimal performance.
- Addresses Non-IID Data Challenges: The research acknowledges the challenge of non-independent and identically distributed (non-IID) data in real-world FL scenarios. FedLayerWise specifically tackles this challenge by adapting weights based on individual client contributions.
- Demonstrates Improved Performance: Experiments show that FedLayerWise outperforms existing FL algorithms like FedAvg [1] and FedAdp [3], especially with skewed non-IID data. This translates to faster convergence and potentially better model performance.

The experiment results on the MNIST dataset demonstrate a reduction in computation rounds of about 40% compared to FedAdp and about 50% compared to FedAvg under highly skewed non-IID data distribution, demonstrating the efficiency gains achieved by FedLayerWise. Overall, this research contributes to the advancement of FL by proposing a novel algorithm that addresses real-world data challenges and improves efficiency in training neural networks while preserving privacy.

The paper is organized as follows. Section II provides an overview of research in FL algorithms. Section III describes our proposed FedLayerWise algorithm and formulas to compute the layer wise adaptive weighting values. Section IV presents the experiment results and evaluation of the proposed method compared with other popular algorithms. The last section provides the conclusion and future work.

II. Related Work

The proliferation of Internet of Things (IoT) devices has prompted increased contemplation on optimizing their use and efficiently managing the generated data. This contemplation has given rise to the concept of FL. Within FL, two critical components merit attention: the communication protocols between clients and the global server and the mechanisms by which the global server integrates updated weights from the clients. IoT devices yield exceptionally rich datasets conducive to model training. However, these datasets often contain sensitive, personally identifiable information. To address this privacy concern, a prevalent approach involves bringing the model to the data through the FedAvg approach [1], where training occurs locally on client devices, and only model weights are transmitted to the central server.

A significant challenge encountered in FL pertains to non-identically and independently distributed (non-IID) data [4], [5], which can decrease the accuracy of trained models by up to 55% [6]. One method is to selectively choose clients and apply reinforcement learning, rewarding higher validation accuracy while penalizing increased rounds, ensuring more efficient model training [7]. Another method is "Agnostic Federated Learning" [8], where the global model is optimized to accept clients with any type of data distribution.

Another challenge, in addition to the existence of non-IID data, pertains to communication. In network communication, the process involves two-way data transmission: uplink communication from clients to the server and downlink communication from the server to clients. Uplink communication is notably slower than downlink communication [9]. Techniques such as structured updates and sketched updates have been instrumental in minimizing the volume of data transmitted during uplink communication to the global server while preserving the accuracy of the training model. These methods have substantially reduced communication costs by orders of magnitude [9].

After model training on the client side is completed, clients transmit their model weights to the global server. Various strategies have been devised to expedite the FL process. [1] Early strategies involved weight aggregation based on the number of samples clients trained on and [1] updating a part of the client model less frequently than the other half. More recent approaches like FOLB [2] and FedAdp [3] utilize correlations between local gradients and global gradients to determine the extent to which a client's model weight influences updates to the global model. These strategies aim to minimize the number of FL rounds required for effective model training.

In this research paper, we utilize a variation of aggregation that relies on the relationship between local gradients and global gradients at each layer. This approach guides the necessary updates to the global model, forming a fundamental aspect of our study.

III. Methodology

In this section, we outline the methodology aimed at minimizing model convergence. Initially, we examine node contribution, determined by the layer-wise gradient of both local and global models. The diversity in data distribution across client devices directly influences the skewness of node contributions. Utilizing this insight, we adaptively assign weights to the global model. To conclude, we present our evaluation plan for this methodology.

A. Layer Wise Agent Node Contribution

In FL, the adjustment of weights to minimize a client's loss function may differ from the modification needed for the global model to decrease overall loss. Additionally, in the presence of a heterogeneous dataset on the client side, the direction tends to deviate from that of the global model [3]. Despite factoring in this consideration and assigning weights based on the angle between the local model's gradient and the global model's [3], certain layers may negatively impact convergence.

To enhance feature extraction at each layer, we propose adaptively modifying the weight of each layer based on the angle between the gradient of the client's model and the global model at that specific layer, employing in (1). Consider an ANN model M with multiple layers, with a matrix of parameters $\mathbf{w}$, which is a concatenation of parameters matrix $\mathbf{w}_n$ at each layer n over the network. Given the loss function $F(\mathbf{w})$, the angle $\theta_{i,n}$ between the gradient of the layer l^{th} at local model k^{th} compared to the same layer in the global model is computed as below:

$$\theta_{k,l} = \arccos\left(\frac{\langle \nabla F(\mathbf{w}_l(t)), \nabla F_k(\mathbf{w}_l(t))\rangle}{\|F(\mathbf{w}_l(t))\|\|F_i(\mathbf{w}_l(t))\|}\right) \quad (1)$$

Here, $\nabla F(w_l(t))$ represents the global model's gradient for the l^{th} layer, and $\nabla F_k(w_l(t))$ signifies the gradient of the k^{th} client model for the same l^{th} layer. Hence, we call this the lawyer-wise node contribution, which is different from the FedAdp method [3] that computes the model-wise gradient difference.

The gradient of layer l^{th} in the global model is calculated as in (2):

$$\nabla F(\mathbf{w}_l(t)) = \sum_{k=1}^{|K|} \left(\frac{n_k}{\sum_{k'=1}^{|K|} n_{k'}}\right) \nabla F_k(\mathbf{w}_l(t)) \tag{2}$$

where n_k is the number of training data samples at client k^{th} in the set K of participating client nodes.

The layer-wise client gradient is calculated as in (3):

$$\nabla F_k(\mathbf{w}_l(t)) = \frac{\mathbf{w}_l(t-1) - w_l(t)}{\eta} \tag{3}$$

where η is the learning rate used during the training. In many FL implementations, gradients are often concealed, and model weights are directly updated. Hence, it is more convenient to compute the gradient for the layer-wise parameter difference.

This layer-wise approach yields all the angles, referred to as instantaneous angles, for each layer of the client's model. To address potential irregularities that could disrupt the instantaneous angles in a specific round, we incorporate the instantaneous angles from the previous round. This calculation ensures the preservation of the generality of the angles, now termed smoothed angles, computed using (4). Here, the calculation is done for the l^{th} layer of k^{th} client's model and $t \geq 1$ represents the current round.

$$\widetilde{\theta}_{k,l}(t) = \frac{t-1}{t} \widetilde{\theta}_{k,l}(t-1) + \frac{1}{t} \theta_{k,l}(t) \tag{4}$$

Similarly, the computation of $\widetilde{\theta}_{i,n}(t)$ follows the calculation in [3], but we quantify the layer-wise gradient angle $\theta_{i,n}$ of each agent in the global round instead of the gradient angle of the whole model. Note that when $t = 1$, only $\theta_{k,l}(t)$ contributes to the value.

B. Layer-wise Adaptive Weighting

This part presents the process of adaptively aggregating the client parameters based on the layer-wise gradient difference between each local model and the global model.

Having obtained the smoothed layer gradient angle $\widetilde{\theta}_{k,l}(t)$, our objective is to amplify the impact of angles closer to 0 compared to those farther away. This prioritization is essential since layers with smoothed angles closer to 0 positively influence global model convergence, and conversely, those farther away have a less favorable impact. To achieve this, the method employs a non-linear mapping utilizing a variant of the Gompertz function as in [3], but for each layer of neuron in the network rather than the whole model.

The modified calculation is depicted in equation 5. Here, α is a hyperparameter that represents the ceiling of the function $f(\widetilde{\theta}_{k,l}(t))$.

$$f(\widetilde{\theta}_{k,l}(t)) = \alpha(1 - e^{-e^{-\alpha(\widetilde{\theta}_{k,l}(t)-1)}}) \tag{5}$$

Layer-wise weighting factor and layer update: Following the completion of all transformations, we have quantified values for each layer, dependent on the angle between the client's and model's gradient. Subsequently, we apply the SoftMax operation to these values to facilitate the update of the final global weight, as outlined in equations (6) and (7). In this context, K_t denotes the set encompassing all clients participating in the learning process at communication round t, and n_k represents the number of samples on which the k^{th} client is trained.

$$\widetilde{\psi}_{k,l}(t) = \frac{n_k e^{f(\widetilde{\theta}_{k,l}(t))}}{\sum_{k'=1}^{|K_t|} n_{k'} e^{f(\widetilde{\theta}_{k',l}(t))}} \tag{6}$$

$$w_l(t) = \sum_{k=1}^{|K_t|} \widetilde{\psi}_{k,l}(t) w_{k,l}(t-1) \tag{7}$$

FedLayerWise Federated Adaptive Weighting Algorithm: Based on these calculations, we present FedLayerWise as in Algorithm 1. The algorithm involves the execution of two primary functions of ClientUpdate (Algorithm 2) and GlobalUpdate (Algorithm 3).

The FedLayerWise procedures are described as follows. Initially, in the first round, the server dispatches randomly initialized global model weights to all participating clients engaged in FL. Subsequently, each client independently trains the model weights utilizing a shared loss and optimizing function, employing consistent hyperparameters. The trained clients transmit back the layer-wise gradients to the server. Upon receiving gradients from all clients, the server aggregates them to compute the lawyer-wise global gradients and update the model weights before dispatching them back to the clients. The main steps of FedLayerWise are:

- Algorithm 1 Line 2: Iterate each communication round t
- Algorithm 1 Lines 3: Iterate each participating node $k \in K$ in parallel with other participating nodes.
- Algorithm 1 Lines 4: Call the ClientUpdate function (Algorithm 2) and send the previous global model $\mathbf{w}(t-1)$ to the current node k. The ClientUpdate function will train the local model $\mathbf{w}_k(t)$ from $\mathbf{w}(t-1)$, then compute and return the list of layer gradient angles from all layers in the local model. These tasks are done in Algorithm 2.
- Algorithm 2 Line 6: At the client node k, initialize the layer-wise gradient list $\mathbf{Grads}_k$ as an empty list.
- Algorithm 2 Lines 7 and 8: Train the local model $\mathbf{w}_k(t-1)$ from the dispatched global model $\mathbf{w}(t-1)$ to obtain the updated client model $\mathbf{w}_k(t)$.
- Algorithm 2 Lines 9 to 12: Loop to consider each layer l of the client model. Then, the algorithm will compute the layer l^{th} gradient $\nabla F_k(\mathbf{w}_l(t))$ using (3)The computed gradient will be concatenated to the layer-wise gradient list $\mathbf{Grads}_k$.
- Algorithm 2 Lines 3: After all layers of $\mathbf{w}_k(t)$ are consider, the layer-wise gradient list $\mathbf{Grads}_k$ is returned to Algorithm 1 for the global update.

- Algorithm 1 Line 6: After all participating client nodes have returned their list of layer-wise gradient values, Algorithm 1 will call the GlobalUpdate to update the global model $\mathbf{w}(t)$. This task is done by Algorithm 3.
- Algorithm 3 Lines 14 to 19: Loop over all L layers in the global model $\mathbf{w}(t)$ to update the parameters $\mathbf{w}_l(t)$ at each layer l^{th}. It calculates the global layer l^{th} gradient $\nabla F(\mathbf{w}_l(t))$ using (2), compute layer-wise gradient angle $\theta_{k,l}(t)$ by (1), and obtain the layer-wise weight factor $\widetilde{\psi}_{k,l}(t)$ through (4, 6, 5). Finally, it update the parameters $\mathbf{w}_l(t)$ of layer l^{th} in the global model $\mathbf{w}(t)$ using (7).
- Algorithm 3 Line 20: After all layers in the global model are updated, the algorithm will concatenate their parameters and return the new updated global model $\mathbf{w}(t)$.

FedLayerWise, while still efficient, requires additional processing compared to FedADP due to its layer-wise approach. The communication overhead for the FedLayerWise algorithm remains the same as other algorithms that send client model weights to the global server. FedLayerWise will be evaluated through multiple experiments and compared with FedAvg [1] and FedAdp [3] algorithms.

Algorithm 1: FedLayerWise Federated Adaptive Weighting

Input: S: an array of clients involved in federated learning
E: Number of epochs
B: Batch Size
T: Total number of rounds
η: Learning rate
L: Number of layers present in the model

Output: Updates the global model for T rounds

1 Initialize a global model $\mathbf{w}(t=0)$ with L layers;
2 **for** $t \leftarrow 1$ **to** $T-1$ **do**
3 **for** *node* $k \in S_k$ *in parallel* **do**
4 $\mathbf{Grads}_k \leftarrow$ ClientUpdate$(k, E, B, \eta, \mathbf{w}(t-1), (t-1), L)$
5 $w(t) \leftarrow$GlobalUpdate$(\mathbf{Grads}_1, \mathbf{Grads}_2 .... \mathbf{Grads}_{|S|}, L)$

IV. Experimental Results

To assess the effectiveness of our Layer Wise Federated Adaptive Weighing algorithm (*FedLayerWise*), we utilized the Flower Python library [10], which furnishes a simulated environment for evaluating FL approaches. The algorithm was tested on the MNIST dataset, introducing non-identical distributions across various clients .

It's crucial to highlight that our algorithm exhibits versatility in handling non-IID data scenarios. The experiments are designed to compare the performance of our algorithm against *FedAdp* and *FedAvg*. Our approach dynamically assigns weights layer-wise to each client's model, contingent on the client layer's contribution towards the global model. We opted for an alpha value of 5 for non-linear mapping as in (5), aligning with the optimal performance observed in *FedAdp* [3]. The experimental parameters that governed the acquisition of results are as follows.

Algorithm 2: ClientUpdate

Input: i: Client Identification index
E: Number of Epochs
B: Batch Size
η: Learning rate
$\mathbf{w}_k(t-1)$: model parameter
t: current round
L: Number of layers present in the model

Output: trains the model and returns the list $\mathbf{Grads}_i$ of gradients of each layer in the model

6 $\mathbf{Grads}_k = \emptyset$;
7 Train the client model for E epochs using stochastic gradient descent with learning rate of η and batch size of B;
8 $\mathbf{w}_k(t) \leftarrow$ updated model weights after training $\mathbf{w}_k(t-1)$;
9 **for** *each layer* $l \in L$ *in the client model* **do**
10 Calculate layer l^{th} gradient $\nabla F_k(\mathbf{w}_l(t))$
11 using (3);
12 $\mathbf{Grads}_k[l] = \nabla F_k(\mathbf{w}_l(t))$ //concatenated layer gradient to the gradient list
13 **return** gradient list $\mathbf{Grads}_k$

Algorithm 3: GlobalUpdate

Input: Local layer-wise gradient list received from the clients involved in the federated learning, $\mathbf{Grads}_1, \mathbf{Grads}_2 .... \mathbf{Grads}_{|S|}$
L: Number of layers present in the model

Output: Aggregates the local updates and returns updated global parameters $\mathbf{w}(t)$ as a set of L layer's parameters $\mathbf{w}_l(t)$

14 **for** *each layer* $l \in L$ *in the model* **do**
15 Calculate the global layer l^{th} gradient $\nabla F(\mathbf{w}_l(t))$ using (2);
16 Calculate layer-wise gradient angle $\theta_{k,l}(t)$ by (1);
17 Calculate and update the smoothed layer-wise angle $\widetilde{\theta}_{k,l}(t)$ by (4);
18 Calculate the layer-wise non-linear factor $f(\widetilde{\theta}_{k,l}(t))$ using (5) and use it to find the layer-wise weight factor $\widetilde{\psi}_{k,l}(t)$ using (6);
19 Update parameters of layer l^{th} in the global model $\mathbf{w}_l(t)$ using (7)
20 **return** $\mathbf{w}(t)$ the concatenation of all $\mathbf{w}_l(t)$

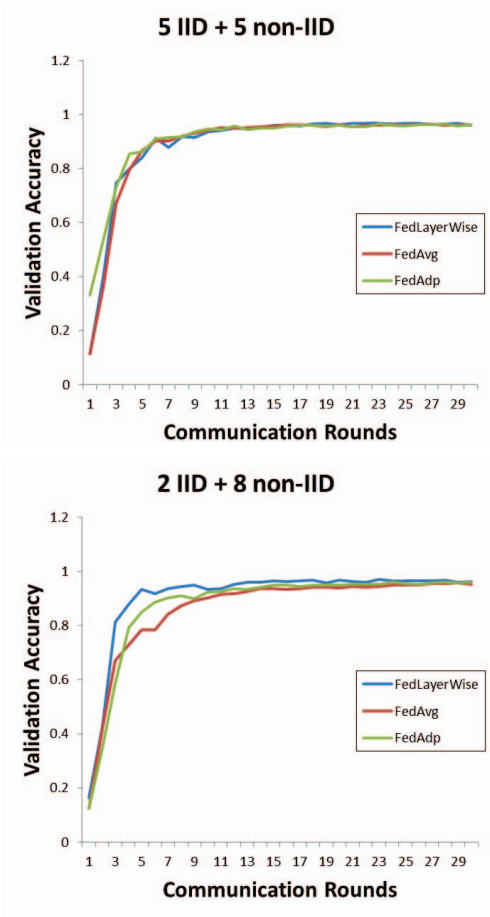

Fig. 1. Validation Accuracy over communication rounds of federated learning with heterogenous data distribution over participating nodes in Convolution Neural Network on MNIST dataset

A. Neural Network Model Architecture:

To ensure an efficient and effective FL approach, we strategically designed a lightweight Convolutional Neural Network (CNN) architecture. This choice balances performance with the computational constraints of training on local mobile nodes. The CNN comprises seven layers, meticulously chosen to extract key features from the data while minimizing trainable parameters (totaling 582,026) with Adam optimizer. Here's a breakdown of the architecture:

- The first Convolutional Layer (5x5, 32 filters) & Max Pooling Layer (2x2): The convolutional layer aims to extract spatial features from the input data. The 5x5 kernel size and 32 filters allow the model to capture essential low-level features. The Max Pooling Layer downsamples the feature maps from the previous layer, reducing the dimensionality of data and computational cost.
- The second Convolutional & Max Pooling: The next layer is a 5x5 Convolutional Layer with 64 filters followed by 2x2 Max Pooling to progressively extract more complex features from the data.
- Two Fully Connected (FC) Layers (1024x512, 512x10): The output from the last Max Pooling is forwarded to two FC layers. The first FC layer transforms the flattened feature maps into a high-dimensional vector, and the second layer performs the final classification task with 10 output neurons (assuming 10 classes).
- Softmax Activation: This activation function ensures the output probabilities sum to 1, providing a clear classification distribution.

The activation function for all convolutional and fully connected layers was ReLU (Rectified Linear Unit), a popular choice for its efficiency and effectiveness in training deep neural networks. To further optimize the training process, we employed the following hyperparameters informed by best practices and potentially adjusted based on the specific dataset:

- Epochs (E) = 1: We opted for a single training epoch (E) considering the constraints of mobile device training and the potential for fine-tuning on the server during aggregation.
- Communication Rounds (T) = 30: This value determines the frequency of communication between devices and the server for model updates.
- Learning Rate & Batch Size: We chose $\eta = 0.005$ based on common practices for CNN training and a batch size of 16 for balance efficiency and memory limitations on mobile devices.
- Local Data Size: Each participating device contributes 600 data samples to the training process

The architecture drew inspiration from established models like FEDAdp, ensuring a foundation for maximum FEDAdp performance compared with our proposed algorithm. Ten (10) client nodes were created in our experiments. Implementation code and data are available upon appropriate request.

B. Obtained Results and Discussion

The experiment results have clear evidence that our proposed algorithm, FedLayerWise, outperformed the other two algorithms. Notably, FedLayerWise demonstrated a significantly faster convergence compared to both FedAdp and FedAvg under scenarios with highly skewed non-IID data distribution. In addition, in the case of a relatively balanced distribution of IID and non-IID data among the clients, all algorithms exhibit commendable performance.

The validation accuracy curves shown in Fig. 1 illustrate the performance of the three algorithms on two training scenarios:

- 5IID+5nonIID: Five client nodes are IID, and the other five are nonIID
- 2IID+8nonIID: Two client nodes are IID, and eight nodes are nonIID. This is highly skewed non-IID data.

The validation loss of these scenarios is illustrated in Fig. 2, while the summary of the communication round to obtain 95% classification accuracy is shown in Table I. Based on these results, we can observe that:

- **FedAvg results:** The obtained results show a steady increase in validation accuracy and a decrease in validation loss over communication rounds. However, its convergence rate might be slower compared to other

TABLE I
NUMBER OF COMMUNICATION ROUNDS REQUIRED BY FEDLAYERWISE, FEDADP AND FEDAVG TO CONVERGE TO 95% ACCURACY FOR MNIST

Fed Methods	Number of communication rounds to converge	
	5 IID- + 5 non-IID	2 IID + 8 non-IID
FedLayerWise	11	12
FedAdp	12	21
FedAvg	12	25

algorithms. FedAvg took about 12 communication rounds to converge on the 5IID+5Non-IID scenario. In the case of 2IID+8Non-IID, FedAvg, it took double the number of rounds (25) to converge.

- **FedAdp results:** This algorithm can address the non-IID data problem better. It might outperform FedAvg in terms of convergence rate, especially for non-IID data scenarios. FedAdp took about 12 communication rounds to converge on 5IID+5Non-IID and took 21 rounds to converge on 2IID+8Non-IID.
- **FedLayerWise results:** FedLayerWise shows the fastest improvement in validation accuracy and the most significant decrease in validation loss compared to the other two algorithms in the 2IID+8Non-IID scenario. FedLayerWise took only 12 communication rounds to converge in 2IID+8Non-IID, which is about 40% less than FedAdp and 50% less than FedAvg. In the 5IID+5Non-IID scenario, FedLayerWise communication rounds about 11, which is similar to FedAdp and FedAvg.

This would suggest that FedLayerWise is more effective in utilizing the information from local models and achieving better overall performance.

V. CONCLUSION

This research investigated Federated Learning (FL) as a privacy-preserving approach for training neural network models. While FL offers advantages like data privacy and scalability, challenges arise due to non-independent and non-identically distributed (non-IID) data distributions in real-world scenarios. To address these challenges, we introduced the *FedLayerWise* algorithm, designed to dynamically allocate weights to each layer of clients' models based on their contributions to each layer in the global model. Through experimentation on the MNIST dataset with different scenarios of IID and non-IID data, we observed that *FedLayerWise* is more robust to non-IID data distributions and can converge faster even in such scenarios. FedLayerWise outperforms existing algorithms like FedAvg and FedAdp, particularly with skewed non-IID data. In the future, we plan to explore the impact of FedLayerWise on different model architectures and datasets beyond MNIST and investigate how FedLayerWise scales with larger numbers of participating devices in the FL network.

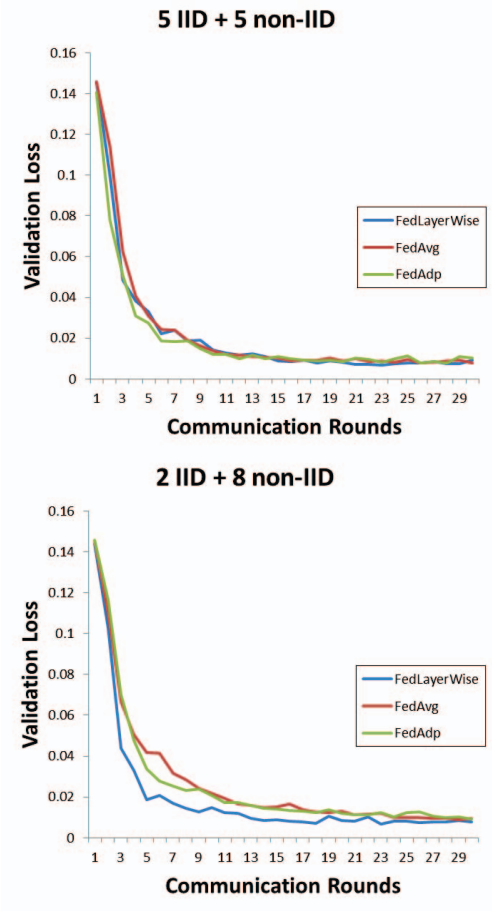

Fig. 2. Validation Loss over communication rounds of federated learning with heterogenous data distribution over participating nodes in Convolution Neural Network. Both the results are on MNIST dataset

REFERENCES

[1] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*. PMLR, 2017, pp. 1273–1282.

[2] H. T. Nguyen, V. Sehwag, S. Hosseinalipour, C. G. Brinton, M. Chiang, and H. V. Poor, "Fast-convergent federated learning," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 1, pp. 201–218, 2020.

[3] H. Wu and P. Wang, "Fast-convergent federated learning with adaptive weighting," *IEEE Transactions on Cognitive Communications and Networking*, vol. 7, no. 4, pp. 1078–1088, 2021.

[4] V. Smith, C.-K. Chiang, M. Sanjabi, and A. S. Talwalkar, "Federated multi-task learning," *Advances in neural information processing systems*, vol. 30, 2017.

[5] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konečnỳ, S. Mazzocchi, B. McMahan *et al.*, "Towards federated learning at scale: System design," *Proceedings of machine learning and systems*, vol. 1, pp. 374–388, 2019.

[6] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-iid data," *arXiv preprint arXiv:1806.00582*, 2018.

[7] H. Wang, Z. Kaplan, D. Niu, and B. Li, "Optimizing federated learning on non-iid data with reinforcement learning," in *IEEE INFOCOM 2020-Conference on computer communications*. IEEE, 2020, pp. 1698–1707.

[8] M. Mohri, G. Sivek, and A. T. Suresh, "Agnostic federated learning," in *International Conference on Machine Learning*. PMLR, 2019, pp. 4615–4625.

[9] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *arXiv preprint arXiv:1610.05492*, 2016.

[10] D. J. Beutel, T. Topal, A. Mathur, X. Qiu, J. Fernandez-Marques, Y. Gao, L. Sani, H. L. Kwing, T. Parcollet, P. P. d. Gusmão, and N. D. Lane, "Flower: A friendly federated learning research framework," *arXiv preprint arXiv:2007.14390*, 2020.