SimFedSwap: Smart Swapping Approach Based on Neural Network Models Similarity for Enhancing Federated Learning

Ali Bozorgzad* Amir Khorsandi Koohanestani*

***Abstract*— Federated learning is a modern framework for distributed training of machine learning models without transferring raw user data. However, a key challenge is statistical heterogeneity, particularly non-IID data, because each user's local data typically represents only a small part of the global distribution. This often causes local models to diverge and reduces the accuracy of the global model. In this paper, we propose SimFedSwap, an intelligent framework that enhances model performance by strategically swapping neural network models between users based on measurable similarity metrics. Unlike random swapping, SimFedSwap selects the least similar models, meaning those trained on the most different data distributions, to maximize the diversity of data each model learns from. We design and evaluate two swapping algorithms in the SimFedSwap framework: greedy swapping and minimum similarity swapping. Experiments on the CIFAR-10, CINIC-10, and FEMNIST datasets demonstrate that our approach accelerates convergence and improves global model accuracy by approximately 1% over random swapping, particularly in realistic, highly non-IID scenarios with a large number of users, such as FEMNIST. Furthermore, all model exchanges are coordinated through the central server, simplifying implementation and offering stronger privacy than direct peer-to-peer swapping.**

***Index Terms*—Federated Learning, Distributed Learning, Non-IID Data, Neural Network Similarity, Statistical Heterogeneity**

I. INTRODUCTION

Machine learning has made remarkable progress in recent years in areas such as image recognition, natural language processing, and recommendation systems [1]. However, the success of these methods directly depends on access to large amounts of training data. In traditional centralized approaches, data is collected from various sources and processed on a central server [2], which brings challenges like high bandwidth usage, processing delays, and especially threats to user privacy [3].

To address these limitations, Federated Learning has been introduced as a distributed framework for training intelligent models without transferring raw data [4]. In this framework, an initial model is sent by a central server to edge devices, and each user updates the model using their local data. Only the updated model parameters (not the raw data) are then sent back to the server [5]. This approach reduces communication costs while preserving user privacy [6].

The FedAvg algorithm, introduced by Google researchers in 2017 [7], is one of the foundational methods in federated learning. However, its performance drops significantly when user data is highly non-IID (non-independent and identically distributed) [8]. This statistical heterogeneity, caused by imbalanced data distribution among users, is a major obstacle to achieving fast and stable convergence of the global model [9].

In real-world settings, each user's data typically represents only a small part of the overall data distribution. For example, in a text prediction system, some users mostly use informal language and emojis, while others focus on scientific terminology. Such data distributions directly contradict the assumption of independence and uniformity used in many classical optimization algorithms [8]. This mismatch leads to divergence of local models, reduced accuracy of the global model, and slow convergence.

To tackle statistical heterogeneity, several solutions have been proposed. Some methods use Multi-Task Learning or Meta Learning to create personalized models for each user [8]. Other approaches, such as FedProx, add a proximal term to the objective function to reduce inconsistencies between local updates and the global model, improving convergence stability in heterogeneous settings [10].

Another line of work aims to increase the diversity of data seen by each model by swapping local models among users [11]. However, these swaps are done randomly and may cause models to relearn similar patterns instead of discovering new ones. This limitation highlights the need for an intelligent strategy to select model pairs with the greatest conceptual differences.

In this paper, we propose a novel method for intelligently swapping local models based on neural network similarity metrics. Instead of random selection, models with the lowest similarity are chosen for swap. This approach is based on the idea that models with lower similarity have likely been trained on more different data. Therefore, swapping them allows each model to encounter patterns it has not seen before. This process accelerates global model convergence and improves its accuracy in non-IID settings.

In summary, the main contributions of this paper are as follows:

- We introduce the SimFedSwap framework, which enables intelligent swapping of user models based on neural network similarity.
- We propose a greedy swapping algorithm that efficiently selects model pairs with the lowest similarity.
- We propose a minimum similarity swapping algorithm that globally identifies optimal model pairs for swapping.

* The authors are with the Department of Electrical and Computer Engineering, Isfahan University of Technology, Iran. E-mail addresses: a.bozorgzad@ec.iut.ac.ir, khorsandi@iut.ac.ir

- We provide a comprehensive evaluation of these methods across diverse datasets and non-IID settings.

The remainder of this paper is organized as follows. Section II reviews related work on non-IID challenges in federated learning. Section III presents our SimFedSwap framework and its two swapping algorithms. Section IV reports experimental results on CIFAR-10, CINIC-10, and FEMNIST datasets. Section V concludes the paper and discusses future directions.

II. RELATED WORK

To address the challenge of statistical heterogeneity or non-IID data in federated learning, many solutions have been proposed in the literature, which we review below.

One early approach is based on data sharing. In this method, the central server provides all users with a small shared dataset so they start training from a common point [12]. This speeds up convergence and improves accuracy. For instance, on datasets with 60,000 samples, using just 5% shared data, can increase accuracy by up to 30% [13]. However, this method requires sharing raw data, which may threaten user privacy [4].

An alternative approach uses Generative Adversarial Networks (GANs) to enhance local data. In this method, each user first uploads a small subset of their data to the server. The server aggregates these samples and trains a GAN, then sends only the generator back to the users. Each user uses this to generate new data and augment their local dataset to help balance class distributions without sharing raw data at scale [14]. This strategy reduces reliance on real data sharing and better preserves privacy. However, the quality of the generated data directly affects model performance and may sometimes amplify noisy or biased patterns.

Another solution that avoids modifying data is intelligent user selection based on data quality. For example, using algorithms like the greedy knapsack method, users with more diverse data receive higher priority to participate in each training round [15]. Deep learning based methods can also predict execution time and select more informative features for training [16]. These approaches optimize computational resources but do not directly address the diversity of data distributions.

At the model level, regularized optimization is a widely used technique. The FedProx algorithm adds a proximal term to the objective function to reduce inconsistencies between local updates and the global model [10]. This improves convergence stability in heterogeneous settings, but it still assumes that a single global model is sufficient for all users an assumption that may not hold in highly heterogeneous environments.

To go beyond this limitation, methods like similarity based clustering have been proposed. In this approach, the server groups users into separate clusters based on the similarity of their local models and trains a customized model for each cluster [17]. This strategy improves accuracy but incurs high communication costs, as it requires frequent exchange of cluster structures between the server and users. Knowledge distillation based methods also reduce data transfer volume and enhance privacy by sending model outputs instead of parameters [18], though they may lose finer details of the learned information.

At the algorithmic level, multi task learning and meta learning have been suggested as ways to handle data diversity. In multi task learning, each user learns a local task, and the server leverages relationships among these tasks to improve the global model [19]. Meta learning increases system flexibility by learning an initial mechanism that quickly adapts to new data [20]. However, these methods often involve high computational complexity and are challenging to implement at scale.

Among these approaches, methods based on swapping local models have drawn special attention. The core idea is that by exchanging trained models among users, each model encounters more diverse data, leading to faster convergence [11]. Yet, existing methods mostly perform these swaps randomly, which may result in exchanging similar models and relearning redundant patterns.

In this context, neural network representation similarity metrics can play a key role. Metrics such as CKA, OSAD, CCA, HSIC, and dCKA quantitatively measure the similarity between two weight matrices [21]. While these metrics have so far been used mainly for analyzing neural networks, we propose using them to guide intelligent model swapping. Instead of random selection, pairs of models with the lowest similarity that is, those trained on the most different data are chosen for exchange. This approach not only increases the diversity of data seen by each model but also accelerates global convergence by reducing overlap in learned patterns, without requiring data sharing or increasing communication costs.

III. OUR APPROACH

A. Similarity-based Federated Swapping (SimFedSwap)

In standard federated learning, after each round, local models are collected from end devices and aggregated into a global model. However, in methods that use a swapping mechanism, this aggregation is not performed in every round. Instead, the server swaps local models between devices at specific steps. In practice, swapping occurs at the end of some rounds, while model aggregation happens at the end of others. These steps are determined by predefined input parameters.

In the FedSwap method, this swapping is done randomly. In our proposed approach, however, the selection and swapping of end devices are based on the similarity between their neural network models. To do this, models must first be compared to measure their similarity. In our proposed SimFedSwap method, the model with the lowest similarity to the current device's model is selected for swapping. This is because high similarity between models reduces the effectiveness of swapping. For a clearer understanding of this structure, refer to Figure 1.

Swapping models between devices with different data increases data diversity and improves the learning process. This

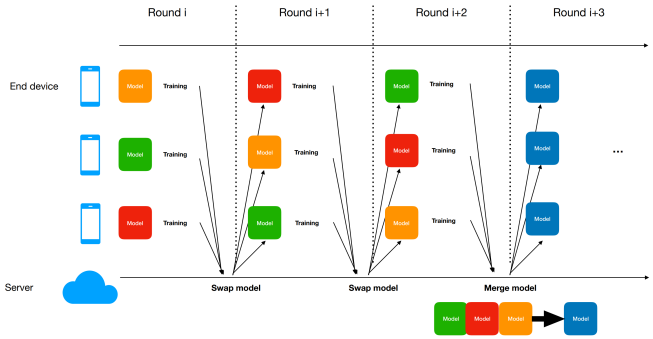

Fig. 1: Federated swapping approach [11].

diversity helps models converge faster and boosts overall performance. It also helps better address the challenges posed by non-IID data.

In this method, all neural network models are first sent to the server. The server then evaluates their similarity and manages the swapping process. This design assumes that end devices have limited hardware and computational resources. All processing and decision making are performed entirely on the server, while end devices only handle lightweight data exchange and updates.

B. Swapping Based on Similarity

When all neural network models are available at the central server, the server must decide which users should swap their models. To make this decision, the models are first compared to measure their similarity. Based on these similarity scores, the server determines which users should exchange their neural network models, or equivalently, which model should be sent to which user.

1) Greedy Swapping (GS): In the greedy approach, a user is first selected at random from the list of available users. The selected user's neural network model is then compared with the models of all other available users to compute pairwise similarity scores. The user whose model exhibits the lowest similarity to the selected model is identified as the swapping partner. The server then exchanges the models of these two users and excludes both from further swapping in the current round.

Once this swap is complete, the same process is repeated for the remaining users: another user is randomly selected, and the same steps are followed. The full pseudocode for this method is provided in Algorithm 1. Additionally, the symbols specific to this algorithm are explained in Table I. This table aim to provide a clear and comprehensive understanding of how the algorithm works and how it is implemented.

2) Minimum Similarity Swapping (MSS): In this method, to achieve the lowest possible similarity among all neural network models, every model must be compared with every other model, and the two models with the smallest similarity are swapped. For example, if there are n models, an $n \times n$ similarity matrix must be created. In this matrix, the similarity between model 1 and model 2 is the same as that between model 2 and model 1. Also, since comparing a model with itself is meaningless, the final matrix takes the form of an upper triangular matrix (excluding the main diagonal). This means only about half of the matrix contains the necessary similarity values for comparison. The resulting matrix looks like this:

$$\begin{bmatrix} \infty & a^{12} & a^{13} & \cdots & a^{1n} \\ \infty & \infty & a^{23} & \cdots & a^{2n} \\ \infty & \infty & \infty & \cdots & a^{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \infty & \infty & \infty & \cdots & \infty \end{bmatrix} \tag{1}$$

First, all pairwise similarities are computed. Then, the smallest similarity value is selected from the matrix. The corresponding row and column indices indicate which two models should be swapped. After this swap, all entries in the rows and columns associated with these two models are set to infinity so they are not selected again in later steps. It is important to update both the row and column for each of the two models.

Algorithm 1: Greedy Swapping (GS)

Function `GreedySwapping()`**:**
  LRS = copy of U_t;
  NS = (length of U_t // 2) * SP;
  for NS *times* **do**
    $RandomIndex$ = random integer between 0 and length of LRS;
    $SCB = LRS[RandomIndex]$;
    remove SCB from LRS;
    Initialize $LstSimilarity$ as an empty list;
    for *each* $RC \in LRS$ **do**
      $Sim =$ `ModelSimilarity`(w^{SCB}, w^{RC});
      Append Sim to $LstSimilarity$;
    end
    $MinSimilarityIndex$ = index of the minimum value in $LstSimilarity$;
    $SCD = LRS[MinSimilarityIndex]$;
    remove SCD from LRS;
    `Swap`(SCB, SCD);
  end
end

TABLE I: Notation for the Greedy Swapping Algorithm

Variable	Description
w	Neural network weights
m	A subset of users
U_t	Unordered set of m at step t
LRS ($LstRemainSwap$)	Remaining swap list
NS ($NumSwaps$)	Number of swaps
SP ($SwapPercentage$)	Control factor for number of swaps
$RandomIndex$	Random index
SCB ($SwapClientBase$)	Source user for swapping
$LstSimilarity$	Similarity list
RC ($RemainClient$)	Remaining user
Sim	Similarity metric between two models
SCD ($SwapClientDest$)	Destination user for swapping

To illustrate, consider Figure 2. Suppose the smallest similarity value is located at row 3 and column 5 of the matrix. In this case, models 3 and 5 should be swapped. After the swap, all values in rows 3 and 5 and columns 3 and 5 are set to infinity, ensuring these two models are excluded from future selections. In the next iteration, the smallest remaining similarity value is chosen from the updated matrix. Since the entries for models 3 and 5 are now infinity, they will not be considered or selected again.

The complete pseudocode for this method is given in Algorithm 2. Additionally, the symbols specific to this algorithm are explained in Table II.

C. Swapping Effects

This section examines the impact of model swapping on network traffic and user privacy. First, we demonstrate that model swapping methods do not increase network traffic compared to traditional approaches and may even reduce it in certain scenarios. Next, we analyze how these swapping operations influence user privacy, assessing the associated risks when models are exchanged directly between users or when a central server mediates the exchange.

1) Impact of Model Swapping on Network Traffic: We will demonstrate that model swapping in FedSwap and SimFedSwap does not add any extra network traffic compared to FedAvg and can even improve traffic efficiency. Assume the total number of users is K, the model size is s, the number of steps between each swapping operation is h_1, and the number of swaps between each global averaging is h_2.

In FedAvg, models are sent to the server and returned after averaging every $h_1 \times h_2$ steps, resulting in a per-round communication cost of $2Ks$.

In FedSwap, models are exchanged directly among users during $h_2 - 1$ steps and are only sent to the server in the final step. The communication cost of this exchange is Ks.

In SimFedSwap, models are sent to and returned from the server in every step, resulting in a per-step cost of $2Ks$.

The communication cost reduction of these methods compared to FedAvg is calculated as follows:

$$\frac{FedAvg - FedSwap}{FedAvg} = \frac{2h_1h_2 - h_2 - 1}{2h_1h_2},$$

$$\frac{FedAvg - SimFedSwap}{FedAvg} = \frac{h_1 - 1}{h_1}$$

With parameters $h_1 = 5$ and $h_2 = 3$, FedSwap reduces network costs by 86.66% and SimFedSwap by 80% (Figure 3).

2) Impact of Model Swapping on Privacy: In FedAvg, users send their locally trained models to the central server for aggregation, with no direct communication between them. All interactions occur exclusively between each user and the server, which simplifies the trust model because users only need to trust the server and not each other.

In model swapping methods, however, privacy risks increase whether the swapping is coordinated by a server or performed directly between users.

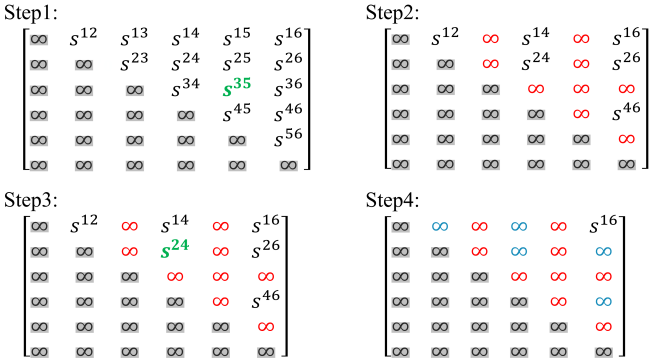

Fig. 2: Steps of minimum similarity swapping.

Algorithm 2: Minimum Similarity Swapping (MSS)

Function `MinSimilaritySwapping()`:
  Initialize $SimArray$;
  L = length of U_t;
  for *each* row *from* 0 *to* L **do**
    for *each* col *from* $(row + 1)$ *to* L **do**
      $Sim =$ `ModelSimilarity`(w^{row}, w^{col});
      Append Sim to $SimArray$;
    end
  end
  NS = $(L$ // $2)$ * SP;
  for NS *times* **do**
    MI = index of minimum value in $SimArray$;
    row, col = find row and col number, based on MI;
    Set all values of $SimArray[row, col]$ to ∞;
    `Swap`(w^{row}, w^{col});
  end
end

TABLE II: Notation for the Minimum Similarity Swapping Algorithm

Variable	Description
$SimArray$	Similarity array
L	Number of users at step t
MI ($MinIndex$)	Index of the min value in the similarity array
row	Row number in similarity matrix
col	Column number in similarity matrix

When a server mediates the swapping process, techniques like Differential Privacy can be applied more consistently, allowing precise control over the amount and placement of added noise. This approach still relies on trust in the server. In contrast, direct model swapping without a server requires each user to independently add noise to their own model before sharing it. This introduces additional complexity and raises the risk of unintended information leakage. For example, in FedSwap, models are exchanged directly between users.

Overall, using a central server as an intermediary generally

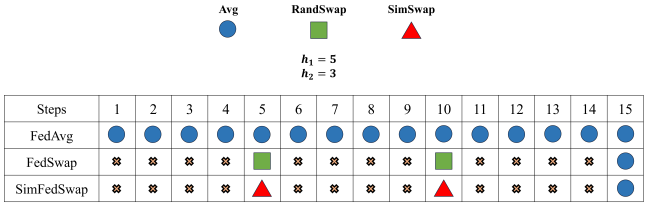

Fig. 3: Impact of model swapping strategy on network traffic.

offers stronger privacy guarantees, whereas direct model exchange between users demands more robust security and privacy-preserving mechanisms.

IV. Experimental Results

In this section, the performance of the proposed SimFedSwap method is evaluated against baseline algorithms such as FedAvg and FedSwap on three benchmark datasets: CIFAR-10, CINIC-10, and FEMNIST. Experiments were conducted under various data distribution settings, including uniform, normal, and a realistic writer-based distribution for FEMNIST, to study the impact of statistical heterogeneity on convergence and accuracy in federated models. Additionally, the two swapping strategies, Minimum Similarity Swapping and Greedy Swapping, were compared within the SimFedSwap framework to assess how different similarity metrics affect the performance of federated learning.

A. CIFAR-10 Dataset

This part presents the experimental results using the CIFAR-10 dataset. Figure 4 compares the SimFedSwap method with other baseline approaches under a uniform data distribution among users.

It should be noted that the FedAvg and FedSwap curves serve as reference baselines and are plotted over the other curves. If a different color appears in the plot, it indicates a performance difference for the corresponding method at that point. This variation may reflect either better or worse performance compared to other methods and can be used as a basis for comparison and analysis.

As shown in Figure 4, swapping based methods demonstrate distinct performance compared to FedAvg. However, these methods achieve similar levels of performance overall. In general, despite minor differences, similarity based methods perform slightly better than FedSwap.

In Figure 5, the same experiment is repeated, but this time using a normal data distribution. Again, swapping based methods show better performance than FedAvg. However, the results among the swapping methods are nearly identical, and no significant difference is observed between them.

B. CINIC-10 Dataset

We now examine the results on the CINIC-10 dataset. Figure 6 illustrates the comparison between the SimFedSwap method and other baseline approaches.

As clearly seen in Figure 6, swapping based methods show different performance compared to FedAvg. However, these methods still operate at a similar performance level, and no major differences are observed among them.

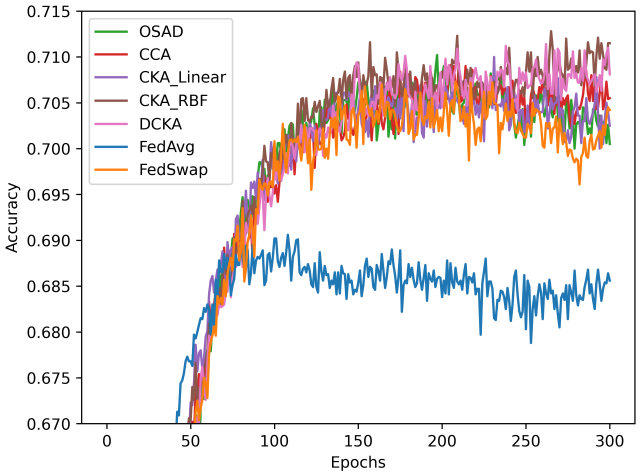

Fig. 4: Accuracy curves on the CIFAR-10 dataset with uniform data distribution.

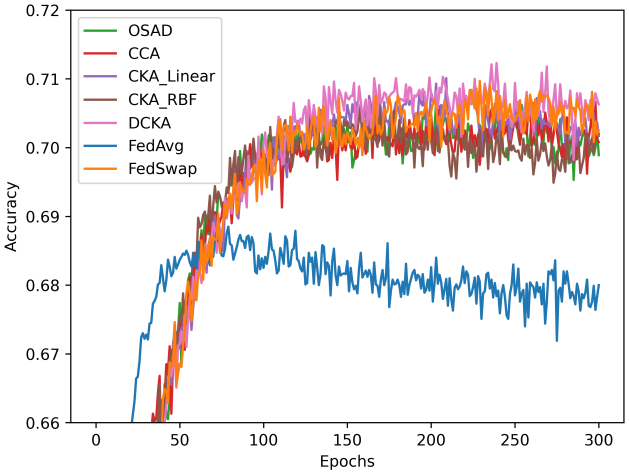

Fig. 5: Accuracy curves on the CIFAR-10 dataset with normal data distribution.

C. FEMNIST Dataset

In this dataset, the number of samples is not equal across classes; different classes contain varying numbers of data points. Figure 7 shows the number of samples per class and how the classes are labeled.

As shown in Figure 7, classes 0 to 9 (representing digits 0 through 9) have significantly more samples than other classes, with each containing approximately 40,000 examples. Among classes 10 to 35 (which correspond to uppercase English letters), classes S and O have the highest number of samples. This pattern likely stems from the data collection process, which aimed to avoid confusion between the letter O and the digit zero, as well as between the uppercase S and its lowercase counterpart in classes 36 to 61.

The FEMNIST dataset by default includes 3,597 users, with data distributed among them. This distribution is uneven both in terms of the number of images per user and the range of classes covered by each user. However, the number of users and the way data is assigned to them can be adjusted

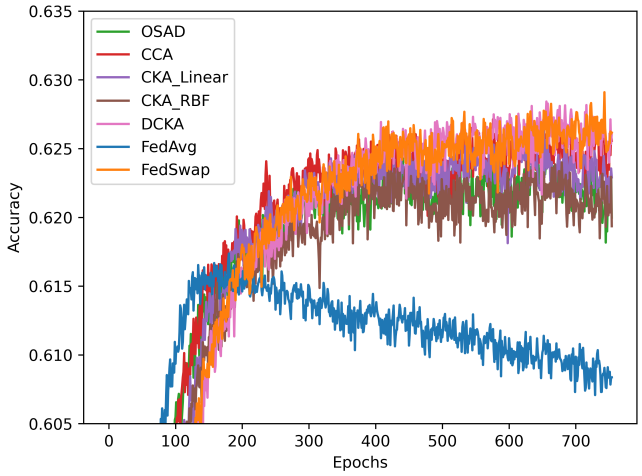

Fig. 6: Accuracy curves on the CINIC-10 dataset.

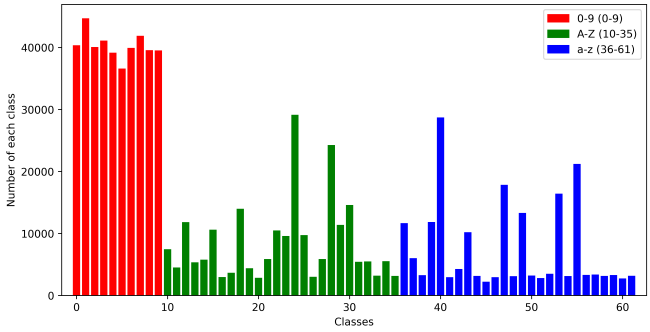

Fig. 7: Number of samples per class and labeling scheme in the FEMNIST dataset.

as needed. We now examine two baseline approaches on the FEMNIST dataset:

In the first approach, the data is partitioned exclusively by class, without regard to the original user assignments. All samples belonging to each class are collected and then distributed among a set number of users according to a specific distribution. This method is referred to as the class based approach or FEMNISTclass.

In the second approach, the original structure of the dataset remains unchanged. The number of users stays at the default value, and each user retains exactly the same data they were originally assigned. This method is known as the writer based approach or FEMNISTwriter.

In the following sections, results for each approach are analyzed separately, including comparisons of SimFedSwap with baseline methods on both the FEMNISTclass and FEMNISTwriter datasets.

1) Class-Based Approach (FEMNISTclass): As seen in Figure 8, swapping based methods exhibit different behavior compared to FedAvg. However, their performance differences remain small and they operate at a similar level overall. It is worth noting that many fluctuations in the plots correspond to the timing of swapping or averaging operations.

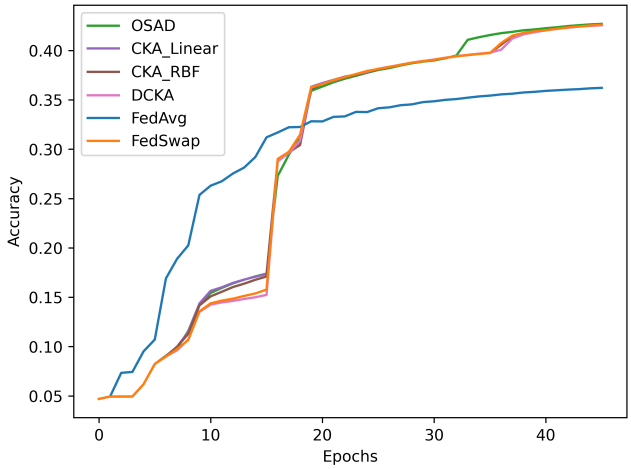

Fig. 8: Accuracy curves on the FEMNISTclass dataset.

2) Writer-Based Approach (FEMNISTwriter): As shown in Figure 9, model-swapping strategies yield performance that differs from the baseline FedAvg. More notably, methods guided by neural network similarity consistently outperform FedSwap. This is the first experiment in our study where similarity-driven approaches show a clear and meaningful improvement.

To evaluate the reliability of this result, we repeated the experiment under a lot of independent initializations. The averaged outcomes across these trials are presented in Figure 10. The performance advantage of similarity-based methods over FedSwap persists, though it is more modest than in the initial run, with an average accuracy gain of approximately 1%.

We also observed that, in earlier experiments, changing the initialization conditions did not lead to significant variations in performance, indicating stable behavior across different runs.

Therefore, we can conclude that as the number of users and their associated data increases, similarity based methods can achieve slightly better performance, though the gain is modest.

It is important to mention that the overall accuracy of around 50 percent on this dataset is tied to the neural network architecture used. Higher accuracy could be achieved by optimizing the network design. However, such optimization would not affect the relative comparison between methods, as all approaches would simultaneously benefit from the improved architecture.

D. Comparison of GS and MSS in SimFedSwap

Figure 11 compares the performance of the Greedy Swapping (GS) and Minimum Similarity Swapping (MSS) strategies within the SimFedSwap algorithm. This experiment was conducted on the FEMNISTwriter dataset.

The results in Figure 11 show that the OSAD similarity metric, when used with the greedy swapping strategy, achieves performance similar to FedSwap. In contrast, the

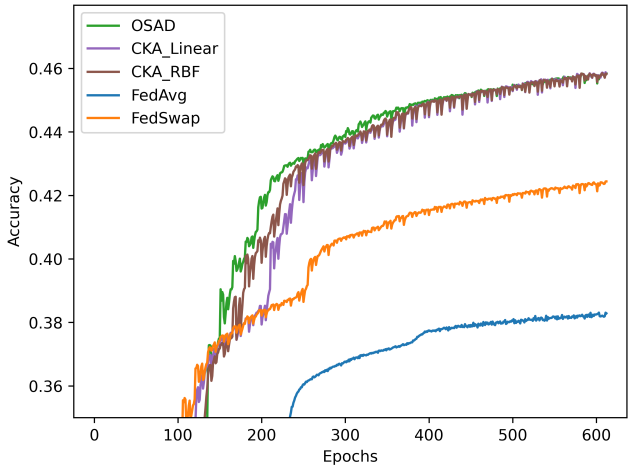

Fig. 9: Accuracy curves from a single run on the FEMNISTwriter dataset.

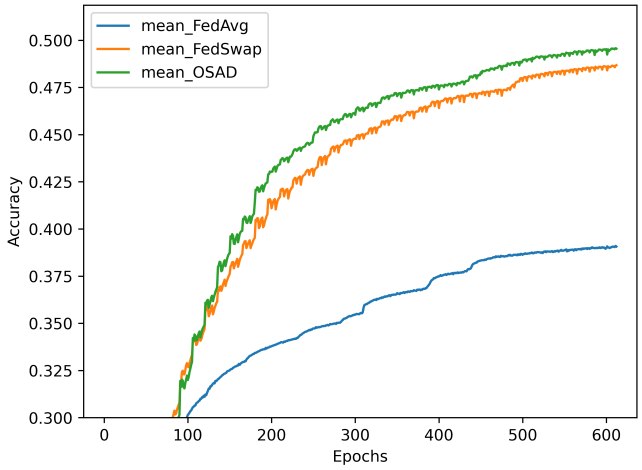

Fig. 10: Accuracy curves averaged over lots of runs on the FEMNISTwriter dataset.

CKA metric, using both linear and Gaussian kernels, gradually improves its performance over time and ultimately yields better results.

V. Conclusion and Future Work

We proposed SimFedSwap, a federated learning method that improves convergence and accuracy under non-IID data by swapping local models based on neural network similarity. Two algorithms were introduced: Greedy Swapping, which iteratively pairs a randomly selected model with its least similar counterpart, and Minimum Similarity Swapping, which globally selects the pair of models with the lowest similarity. Experiments on CIFAR-10, CINIC-10, and FEMNIST show that both strategies enhance performance over random swapping, with consistent gains, especially in realistic, large-scale settings like FEMNISTwriter, where SimFedSwap achieves up to 1 percent higher accuracy. The approach requires no extra data sharing, preserves privacy via server-mediated swapping, and reduces communication costs compared to FedAvg.

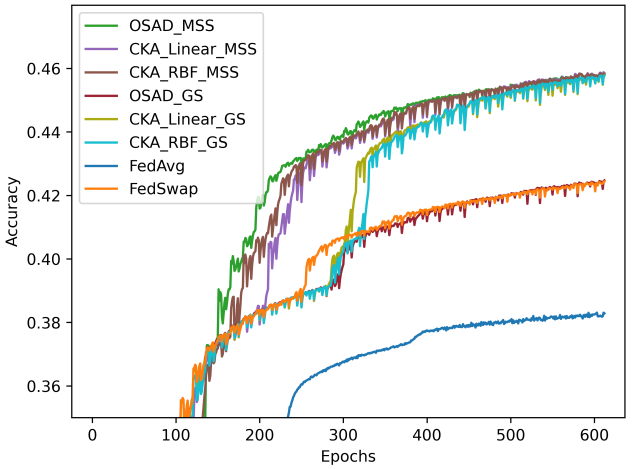

Fig. 11: Accuracy curves comparing GS and MSS on the FEMNISTwriter dataset.

Future research could enhance similarity-based swapping in several directions: (1) using multiple similarity metrics simultaneously and selecting swaps based on the overall minimum across all metrics; (2) replacing the uniform layer-wise average with a weighted average that reflects the relative importance of each layer; and (3) exploring cross-layer similarities between non-corresponding layers to capture deeper structural relationships and improve the accuracy of similarity estimation.

References

[1] A. M. Elbir, S. Coleri, A. K. Papazafeiropoulos, P. Kourtessis, and S. Chatzinotas, "A family of hybrid federated and centralizedlearning architectures in machine learning," *IEEE Transactions on Cognitive Communications and Networking*, 2022.

[2] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, and J. Zhang, "Edge intelligence: Paving the last mile of artificial intelligence with edge computing," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1738–1762, 2019.

[3] M. Talaei and I. Izadi, "Adaptive differential privacy in federated learning: A priority-based approach," *arXiv preprint arXiv:2401.02453*, 2024.

[4] X. Ma, J. Zhu, Z. Lin, S. Chen, and Y. Qin, "A state-of-the-art survey on solving non-iid data in federated learning," *Future Generation Computer Systems*, vol. 135, pp. 244–258, 2022.

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

[6] D. R. Brendan McMahan, "Federated learning: Collaborative machine learning without centralized training data," https://research.google/blog/federated-learning-collaborative-machine-learning-without-centralized-training-data/, 06 Apr 2017, [Accessed: 20 Mar 2022].

[7] 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.

[8] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE signal processing magazine*, vol. 37, no. 3, pp. 50–60, 2020.

[9] J. Konečnỳ, B. McMahan, and D. Ramage, "Federated optimization: Distributed optimization beyond the datacenter," *arXiv preprint arXiv:1511.03575*, 2015.

[10] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," *Proceedings of Machine learning and systems*, vol. 2, pp. 429–450, 2020.

[11] T.-C. Chiu, Y.-Y. Shih, A.-C. Pang, C.-S. Wang, W. Weng, and C.-T. Chou, "Semisupervised distributed learning with non-iid data for aiot service platform," *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 9266–9277, 2020.

[12] 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.

[13] L. Collins, H. Hassani, A. Mokhtari, and S. Shakkottai, "Exploiting shared representations for personalized federated learning," in *International conference on machine learning*. PMLR, 2021, pp. 2089–2099.

[14] E. Jeong, S. Oh, H. Kim, J. Park, M. Bennis, and S.-L. Kim, "Communication-efficient on-device machine learning: Federated distillation and augmentation under non-iid private data," *arXiv preprint arXiv:1811.11479*, 2018.

[15] A. Taïk, H. Moudoud, and S. Cherkaoui, "Data-quality based scheduling for federated edge learning," in *2021 IEEE 46th Conference on Local Computer Networks (LCN)*. IEEE, 2021, pp. 17–23.

[16] Y. Zeng, X. Wang, J. Yuan, J. Zhang, and J. Wan, "Local epochs inefficiency caused by device heterogeneity in federated learning." *Wireless Communications & Mobile Computing*, 2022.

[17] A. Ghosh, J. Hong, D. Yin, and K. Ramchandran, "Robust federated learning in a heterogeneous environment," *arXiv preprint arXiv:1906.06629*, 2019.

[18] S. Itahara, T. Nishio, Y. Koda, M. Morikura, and K. Yamamoto, "Distillation-based semi-supervised federated learning for communication-efficient collaborative training with non-iid private data," *IEEE Transactions on Mobile Computing*, vol. 22, no. 1, pp. 191–205, 2021.

[19] L. Corinzia, A. Beuret, and J. M. Buhmann, "Variational federated multi-task learning," *arXiv preprint arXiv:1906.06268*, 2019.

[20] Y. Jiang, J. Konečnỳ, K. Rush, and S. Kannan, "Improving federated learning personalization via model agnostic meta learning," *arXiv preprint arXiv:1909.12488*, 2019.

[21] S. Kornblith, M. Norouzi, H. Lee, and G. Hinton, "Similarity of neural network representations revisited," in *International conference on machine learning*. PMLR, 2019, pp. 3519–3529.