Received XX Month, XXXX; revised XX Month, XXXX; accepted XX Month, XXXX; Date of publication XX Month, XXXX; date of current version 19 September, 2024.

Digital Object Identifier 10.1109/OJCOMS.2024.011100

SimProx: A Similarity-Based Aggregation in Federated Learning with Client Weight Optimization

Ayoub El-Niss[1], Ahmad Alzu'bi[1] *(Senior Member, IEEE)*, Abdelrahman Abuarqoub[2] *(Member, IEEE)*, Mohammad Hammoudeh[3] *(Senior Member, IEEE)*, Ammar Muthanna[4] *(Senior Member, IEEE)*

[1]Department of Computer Science, Jordan University of Science and Technology, Irbid, Jordan
[2]Department of Applied Computing, Cardiff School of Technologies, Cardiff, UK
[3]Department of Computing and Mathematics, Manchester Metropolitan University, Manchester, UK
[4]Department of Telecommunication Networks and Data Transmission, St. Petersburg State University of Telecommunication, St. Petersburg, Russia

CORRESPONDING AUTHOR: Ahmad Alzu'bi (e-mail:agalzubi@just.edu.jo)

This work was supported by St. Petersburg State University of Telecommunications. The studies at St. Petersburg State University of Telecommunications were supported by the Ministry of Science and High Education of the Russian Federation by the grant 075-15-2022-1137

ABSTRACT Federated Learning (FL) enables decentralized training of machine learning models across multiple clients, preserving data privacy by aggregating locally trained models without sharing raw data. Traditional aggregation methods, such as Federated Averaging (FedAvg), often assume uniform client contributions, leading to suboptimal global models in heterogeneous data environments. This article introduces SimProx, a novel FL approach for aggregation that addresses heterogeneity in data through three key improvements. First, SimProx employs a composite similarity-based weighting mechanism, integrating cosine and Gaussian similarity measures to dynamically optimize client contributions. Then, it incorporates a proximal term in the client weighting scheme, using gradient norms to prioritize updates closer to the global optimum, thereby enhancing model convergence and robustness. Finally, a dynamic parameter learning technique is introduced, which adapts the balance between similarity measures based on data heterogeneity, refining the aggregation process. Extensive experiments on standard benchmarking datasets and real-world multimodal data demonstrate that SimProx significantly outperforms traditional methods like FedAvg in terms of accuracy. SimProx offers a scalable and effective solution for decentralized deep learning in diverse and heterogeneous environments.

INDEX TERMS Federated Learning, Decentralized Network, Weighted Aggregation, Data Heterogeneity, Deep Learning, Multimodal Classification

I. INTRODUCTION

FEDERATED Learning (FL) [1] has recently gained significant attention since it attempts to train Machine Learning (ML) models with a decentralized approach, addressing major concerns with data security and privacy. By enabling multiple clients to collaboratively contribute to training ML models without sharing their raw data, FL maintains data confidentiality while utilizing the collective power of distributed datasets. Therefore, federated learning plays an increasingly important role in maintaining privacy and attaining good model performance results as the amount of data generated globally keeps growing [2], [3].

While FL offers many advantages, it faces considerable challenges in building a generalized and robust global model from locally trained models. The essence of FL lies in aggregating local updates from distributed clients to create a unified global model without sharing raw data. The more realistic challenging scenario of heterogeneous FL remains a key challenge despite the tremendous progress made in homogeneous FL, where clients share similar network architectures and analyze similar data distributions [4], [5].

In most large-scale, real-world applications, clients exhibit considerable differences in data distributions, communication networks, and model structures, leading to four primary types of heterogeneity.

Firstly, statistical heterogeneity [6] arises from varying data distributions across clients participating in the FL process. Secondly, model heterogeneity [7] occurs when different clients employ distinct models, which may differ in architecture, size, customizations, learning rates, and hyper-parameters. Researchers have attempted to address this issue using various approaches, including model mapping [8], knowledge distillation [9], ensemble learning [10], and meta learning [11]. Thirdly, communication heterogeneity [12] emerges when clients operate under diverse network environments. Lastly, device heterogeneity [13] arises from the varying storage and computation capabilities of devices among participants, which can lead to faults and inactivation of some nodes, known as stragglers [14], [15]. To mitigate these challenges, several methods have been developed, including asynchronous aggregation [16], [17], which enables clients to upload their local updates in a staggered manner.

Many studies have concentrated on FL aggregating techniques that mitigate the impact of statistical heterogeneity on building a global model on the server side [18]–[20], starting with FedAvg [21] where a group of clients is randomly selected at each round of training for aggregation. During the aggregation process, the parameters of each client are weighted and averaged to produce a global model. Despite that FedAvg can handle Non-Independent and Identically Distributed (Non-IID) [22] data to some extent, many research investigations [23], [24] have demonstrated that a degradation in the accuracy of FL is practically certain when dealing with Non-IID or heterogeneous data.

This paper addresses the non-IID issues from a novel perspective based on the following observation: *if two models training on different (skewed) datasets have learned the same information, then this particular information is crucial for producing a generalized model*. Consequently, we propose a new aggregation technique based on the similarity information of client models. Unlike existing approaches that utilize similarity for clustering, the proposed technique, Similarity-Proximal (SimProx), employs a similarity matrix to calculate weights for clients and performs weighted sum aggregation. It incorporates a proximal term that considers the gradient norms as a proxy for the client's contribution to the global model update. This term indicates how much the client model has changed during the current round of FL. The specific contributions of this work are as follows:

1) Propose a method to calculate the similarity matrix using both Gaussian and cosine similarity, which helps the global model to learn from the most relevant and complementary client updates.
2) Include a proximal term into the deep FL aggregation to complement the client similarity information captured by the similarity matrix, stimulating the global model to learn more from the client updates that are closer to the global optimum. Additionally, we apply a dynamic calculation of the lambda learning parameter based on the heterogeneity of learnt data.
3) Evaluate the performance of the proposed method on real-world multimodal data, textual and visual representation, and two versions of CIFAR testbed.

The remaining part of this paper is structured as follows: Section II introduces the general task of FL and then reviews the related work regarding similarity-based aggregation in heterogeneous settings. Section III presents the methodology adopted in this study. Section IV presents the experimental setups and discusses the model's results in the context of previous approaches. Section V presents the evaluation of real-world multimodal data. Finally, Section VI concludes the paper and highlights the main finding.

II. TASK FORMULATION

In this section, we provide a comprehensive review of various aggregation approaches in FL. To facilitate a thorough understanding of these approaches, we first establish a common task notation.

A. Federated Learning

Federated learning is a distributed machine learning approach that enables model training across multiple decentralized devices or servers holding local data samples, without exchanging their data. Unlike traditional centralized machine learning methods that require pooling data to a central server, FL maintains data privacy and security by keeping data on local devices.

Given the list of preliminaries shown in Table 1, consider a FL system with C clients, each holding a local dataset $\mathcal{D}_i$ where $i \in \{1, 2, \ldots, C\}$. The objective is to train a global model w by aggregating local models w_i trained on the local datasets $\mathcal{D}_i$. In order to achieve the primary objectives of FL, the following factors need to be maintained:

1) **Data Privacy:** Preserve the privacy of local data by ensuring that raw data remains on the client's device.
2) **Communication Efficiency:** Minimize the communication overhead between clients and the central server to make the system scalable and efficient.
3) **Robustness to Heterogeneity:** Handle non-IID data distributions across clients, ensuring that the global model generalizes well despite variations in local data.
4) **Decentralized Model Training:** Enable collaborative model training across multiple clients without the need for centralized data aggregation.

Improving aggregation methods is crucial for FL, especially when dealing with non-IID data. In FL, data is distributed across multiple clients, often with significant differences in data distribution and quality. Traditional aggregation methods, such as simple averaging, may not adequately address these disparities, leading to suboptimal

TABLE 1: FL task preliminaries.

Notation	Description
C	number of participating clients
C_i	i-th client
D_i	private dataset of client C_i
N_i	number of samples in dataset D_i
N	total number of samples across all clients
w_i	model weights of client i
$f(w_i)$	learned local network model of client C_i
$w_central$	centralized model parameters
t	epoch number in the FL process
w_t	global model parameters at epoch t
d_{ij}	Euclidean distance between w_i and w_j
σ	average distance between all clients
s_{ij}	cosine similarity between w_i and w_j
s_g	Gaussian similarity between w_i and w_j
λ	hyperparameter controlling trade-off between s_{ij} and s_g
S	client similarity matrix
α_i	weight of client i

model performance [23]. Non-IID data poses additional challenges because the variations in data can cause local models to diverge significantly, complicating the aggregation process [7]. Advanced aggregation methods that consider the underlying data distribution and client similarities can help mitigate these issues, ensuring more robust and accurate global models. By developing and implementing more sophisticated aggregation techniques, FL systems can achieve better generalization, resilience to client variability, and ultimately, more effective and reliable AI solutions in decentralized environments [13], [22].

Several research studies have attempted to improve data aggregation from the client's side, particularly for non-IID data, which typically focus on either local training or the aggregation process. However, Our study's objective is to improve the aggregation process, offering a more adaptive and robust solution to the challenges of FL in diverse real-world settings.

B. Aggregation Algorithms

FedAvg [21] is a fundamental approach in FL, where model updates are aggregated across clients through a weighted average. FedAvg framework involves four key steps in each iteration: the server distributes the global model w_t to all participating clients, each client i updates their local model w_i by performing Stochastic Gradient Descent (SGD) on their local data $\mathcal{D}i$, the clients send their updated local models $w_i^{(t+1)}$ back to the server, and the server aggregates the local models by averaging them to produce the new global model, which is defined as follows:

$$w^{t+1} = \frac{1}{C}\sum_{i=1}^{C} w_i^{(t+1)} \tag{1}$$

where C is the number of clients.

However, FedAvg assumes that the data distributions across clients are identical and independent, which is often not the case in real-world scenarios [25]. To address this limitation, FedProx [26] extends FedAvg by introducing a proximal term to handle heterogeneous data. The objective function for each client includes a proximal term to maintain proximity to the global model:

$$\mathcal{L}_i^{\text{FedProx}}(w) = \mathcal{L}_i(w) + \frac{\mu}{2}|w - w_t|^2 \tag{2}$$

where μ is a regularization parameter. This approach has been shown to improve the robustness of FL systems in the presence of non-IID data.

Another approach to improve the efficiency of FL is FedDyn [27], which uses dynamic regularization to reduce the variance between local and global models. The objective function for each client includes a dynamic regularization term as follows:

$$\mathcal{L}_i^{\text{FedDyn}}(w) = \mathcal{L}_i(w) + \lambda|w - w_t|^2 \tag{3}$$

where λ is a dynamic regularization parameter. FedDyn has been demonstrated to achieve improved communication efficiency in FL systems.

More sophisticated similarity-based aggregation approaches were introduced. SimAgg was introduced by Khan et al. [28] to improve model aggregation by weighting client model contributions based on how similar they are to the global model. This method demonstrates significant improvements in settings with highly heterogeneous data. Moreover, Wu and Wang [29] proposed an adaptive weighting approach aimed at speeding up convergence in FL by dynamically adjusting the contribution of each client's update based on its relevance to the global model.

In addition to these approaches, Modeling Overlapping Neighborhoods (MOON) [30] focuses on better client aggregation strategies by considering overlapping neighborhoods. The loss function includes a contrastive loss term to improve model consistency across overlapping data distributions:

$$\mathcal{L}_i^{\text{MOON}}(w) = \mathcal{L}_i(w) + \gamma\sum j \in \mathcal{N}(i)|w_i - w_j|^2 \tag{4}$$

where $\mathcal{N}(i)$ represents the neighborhood of client i and γ is a weighting parameter. This approach has been shown to improve the robustness of FL systems in scenarios with overlapping data distributions.

Clustered-based FL aggregation [31], [32] is another approach that involves grouping clients into clusters based on the similarity of their data distributions. Each cluster independently trains a local model, and the global model is formed by aggregating the cluster models:

$$w_{t+1}^{\text{cluster}} = \frac{1}{|C_k|}\sum_{i\in C_k} w_i^{(t+1)} \tag{5}$$

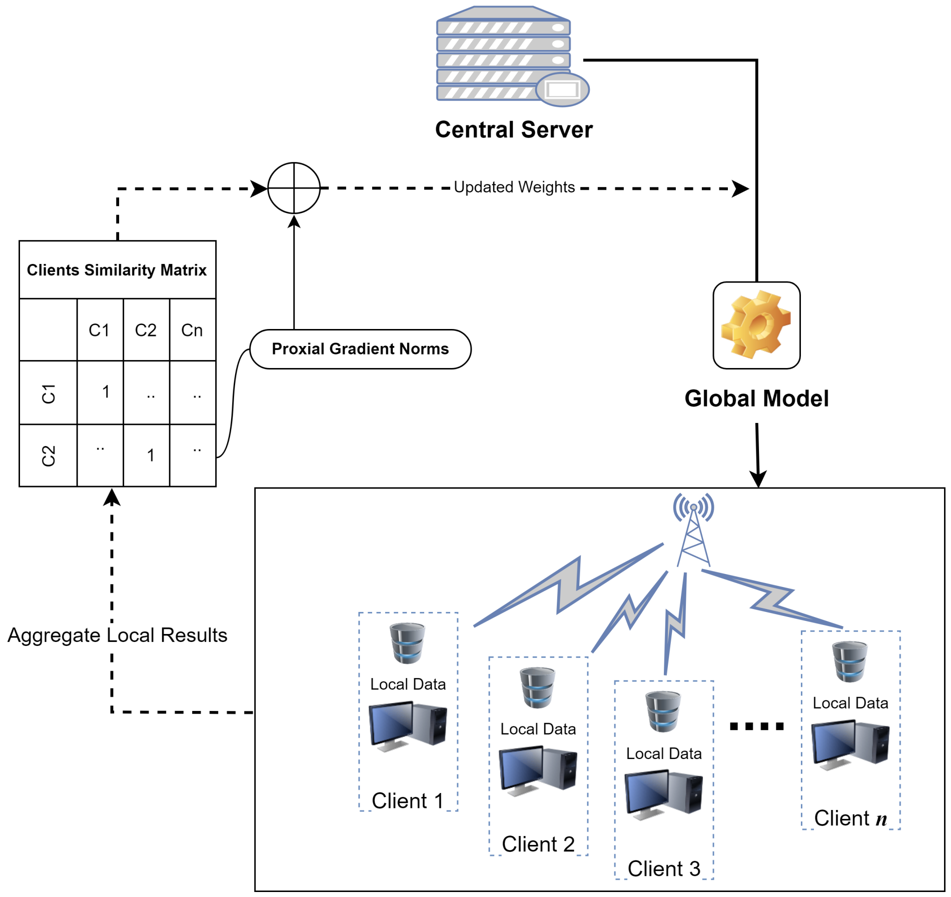


FIGURE 1: The general pipeline of the proposed similarity-based FL aggregation approach.

where C_k is the k-th cluster and $|C_k|$ is the number of clients in cluster k. Hierarchical FL [33] introduces an additional layer of aggregation, where local models are first aggregated at the cluster level and then at the global level:

$$w_{t+1} = \frac{1}{M} \sum_{k=1}^{M} w_{t+1}^{\text{cluster}_k} \tag{6}$$

where M is the number of clusters.

However, our proposed method addresses the gap in existing FL approaches by introducing a similarity-based weighting mechanism to enhance model aggregation. This approach involves computing the average distance between client models to understand their diversity, calculating a combined similarity score using cosine and Gaussian similarity, determining and normalizing client weights, and aggregating client models using the similarity-weighted scores. Additionally, a proximal term with dynamic parameter learning is included to improve the robustness of FL systems, particularly in scenarios with highly heterogeneous data.

III. METHODOLOGY

In this work, we propose a novel FL approach that incorporates client similarity information to improve the overall model performance. Figure 1 shows the generic pipeline of the proposed similarity-based aggregation of FL results from the client side to the server side. A set of clients contribute to the learning procedure performed on the global model broadcasted by a Central server. The results and weights of each single client trained on its local data are aggregated into a similarity matrix that indicates the similarity-weighted scores to understand the diversity of clients' models.

A proximal term with dynamic parameter learning is also involved in the learning procedure, providing the final weighted sum of results aggregated from the clients in each training round. This process is performed over many rounds to improve the convergence and robustness of the FL paradigm. We first establish the baseline distance calculation between clients, followed by detailing each step of the proposed approach in the subsequent subsections.

Let $\mathcal{C} = 1, 2, \ldots, m$ be the set of clients, and let $\mathbf{w}_i \in \mathbb{R}^d$ be the model weights of client i, where d is the dimensionality of the model. To quantify the overall similarity between clients, we define the average distance between clients as:

$$\sigma = \frac{1}{\binom{m}{2}} \sum_{i=1}^{m} \sum_{j=i+1}^{m} \|\mathbf{w}_i - \mathbf{w}_j\|_2 \tag{7}$$

where $\|\cdot\|_2$ denotes the Euclidean norm. This average distance metric captures the diversity of the client models, with a smaller value indicating that clients are more similar, and a larger value indicating that clients are more diverse.

A. Client Similarity Matrix

The core of the proposed aggregation process relies on calculating the similarity between the clients' results. This section explains how Gaussian and Cosine functions are used to determine the similarity matrix, incorporating a dynamically learned hyperparameter to manage the trade-off between similarity scores.

1) Gaussian Similarity

The Gaussian similarity term in the client similarity matrix $\mathbf{S}$ is defined as:

$$S_{ij}^{\text{Gaussian}} = \exp\left(-\frac{\|\mathbf{w}_i - \mathbf{w}_j\|_2^2}{2\sigma^2}\right) \tag{8}$$

This term captures the Euclidean distance between the client model weights $\mathbf{w}_i$ and $\mathbf{w}_j$, normalized by the average distance σ between all client models. The key intuition behind the Gaussian similarity is that clients with more similar model weights, as measured by the Euclidean distance, are likely to provide more relevant and complementary updates to the global model. The Gaussian function is used to map the Euclidean distances to similarity scores, with smaller distances resulting in higher similarity scores.

2) Cosine Similarity

The cosine similarity term in the client similarity matrix $\mathbf{S}$ is defined as:

$$S_{ij}^{\text{Cosine}} = \frac{\mathbf{w}_i^\top \mathbf{w}_j}{\|\mathbf{w}_i\|_2 \|\mathbf{w}_j\|_2} \tag{9}$$

This term captures the angular difference between the client model weights $\mathbf{w}_i$ and $\mathbf{w}_j$, as measured by the cosine of the angle between them. The cosine similarity provides a complementary perspective to the Gaussian similarity. While the Gaussian similarity focuses on the Euclidean distance between the model weights, the cosine similarity captures the alignment or "direction" of the model weights.

Clients with more similar model directions, as indicated by a higher cosine similarity score, are likely to provide updates that are more aligned with the global model's learning objective. These updates can help the global model converge faster and achieve better generalization performance. The cosine similarity is invariant to the scaling of the model weights, as it only depends on the relative direction of the vectors.

3) Hybrid Similarity Measure

By combining the Gaussian similarity and the cosine similarity, the algorithm can capture both the Euclidean distance and the angular difference between the client models, providing a more comprehensive assessment of client similarity. This hybrid similarity measure helps the global model learn from the most relevant and complementary client updates, leading to improved performance and faster convergence. We merge the Gaussian similarity and cosine similarity into a unified similarity matrix $\mathbf{S}$ using a hyperparameter $\lambda \in [0, 1]$ to control the trade-off between the two similarity measures:

$$S_{ij} = \lambda S_{ij}^{\text{Cosine}} + (1 - \lambda) S_{ij}^{\text{Gaussian}} \tag{10}$$

We incorporate a dynamic lambda adjustment mechanism to adaptively control the balance between cosine and Gaussian similarity measures. The average cosine similarity between the client models and the global model is calculated at every round. If this average falls below a predefined heterogeneity threshold, indicating high client diversity, The lambda value is proportionally reduced to the ratio of average similarity to the threshold. Otherwise, the base lambda value is maintained. Formally, let s_i be the cosine similarity between the i-th client model and the global model, and $\bar{s} = \frac{1}{N}\sum_{i=1}^{N} s_i$ be the average similarity. Given a base lambda λ_0 and heterogeneity threshold τ, we adjust lambda as:

$$\lambda = \begin{cases} \lambda_0 \cdot \left(\frac{\bar{s}}{\tau}\right) & \text{if } \bar{s} < \tau \\ \lambda_0 & \text{otherwise} \end{cases}$$

This adaptive approach allows λ to adjust its behavior based on the current level of heterogeneity in the FL system, potentially improving convergence and performance in extreme heterogeneity scenarios. As a result, this hybrid similarity measure allows the algorithm to capture both the Euclidean distance and the angular difference between the client models, providing a more comprehensive assessment of client similarity.

B. Proximal Gradient Norms

In addition to the client similarity, we also consider the gradient norms $\mathbf{g} \in \mathbb{R}^m$ as a proxy for the client's contribution to the global model update. The gradient norm g_i is defined as the Euclidean norm of the difference between the client's current model weights $\mathbf{w}_i$ and its previous model weights $\mathbf{w}_i^{\text{prev}}$:

$$g_i = \|\mathbf{w}_i - \mathbf{w}_i^{\text{prev}}\|_2 \tag{11}$$

The gradient norm g_i serves as an indicator of how much the client model has changed during the current round of FL. Clients with smaller gradient norms are likely to have model updates that are closer to the global optimum, as their current model is already well-aligned with the global model.

Intuitively, this proximal term encourages the global model to learn more from the client updates that are closer to the global optimum, as they are likely to be more beneficial for the overall model performance. Clients with smaller gradient norms are essentially providing "finer-tuned" updates that can help the global model converge faster and achieve better generalization.

This proximal term complements the client similarity information captured by the similarity matrix $\mathbf{S}$. Together, these two components allow the algorithm to identify the most relevant client updates and incorporate them more effectively into the global model update, leading to improved performance and faster convergence of the FL process.

C. Client Weights

The client weights $\boldsymbol{\alpha} \in \mathbb{R}^m$ are calculated as follows:

$$\alpha_i = \exp(-g_i)\left(1 + \frac{\sum_{j\neq i} S_{ij}}{m-1}\right) \tag{12}$$

This formulation combines two key components: the gradient norms $\mathbf{g}$ and the client similarity matrix $\mathbf{S}$. By combining the gradient norm term and the client similarity term, the algorithm assigns higher weights to clients that have both smaller gradient norms (indicating updates closer to the global optimum) and higher average similarity to the rest of the client population (indicating more relevant and complementary updates). This dual consideration of the client's contribution and similarity allows the global model to learn more effectively from the most valuable client updates, leading to improved performance and faster convergence.

The mathematical properties of the individual components, such as boundedness, monotonicity, and normalization, ensure that the client weight calculation is well-behaved and provides a principled way to prioritize the most relevant client updates during the FL process.

After calculating the client weights $\boldsymbol{\alpha}$ using the formulation in Equation (6), we perform an additional normalization step:

$$\boldsymbol{\alpha} \leftarrow \boldsymbol{\alpha} / \sum_{i=1}^{m} \alpha_i \tag{13}$$

Then a softmax function is applied to the weights to provide more evenly weight distribution. This normalization ensures that the client weights sum up to 1, i.e., $\sum_{i=1}^{m} \alpha_i = 1$. The normalization step serves two important purposes, which are proper weighting and interpretability.

On one hand, normalizing the client weights ensures that the weighted sum of the client updates $\mathbf{w}_i$ is a valid model update, as the weights are properly scaled to sum up to 1. This allows the global model $\mathbf{w}^{\text{global}}$ to be updated as a convex combination of the client updates, which is a necessary condition for the FL algorithm to converge. On the other hand, the normalized client weights $\boldsymbol{\alpha}$ can be interpreted as the relative importance or contribution of each client to the global model update. This provides a clear and intuitive way to understand the role of different clients in the FL process.

Algorithm 1 SimProx

```
1: procedure FederatedLearning($\mathbf{w}^{\text{global}}$, $\mathbf{w}^{\text{client}}$, $\mathbf{w}^{\text{prev}}$)
2:     $\mathbf{S} \leftarrow$ ComputeClientSimilarityMatrix($\mathbf{w}^{\text{client}}$)    ▷ Compute the client similarity matrix
3:     $\boldsymbol{\alpha} \leftarrow$ ComputeClientWeights($\mathbf{w}^{\text{client}}, \mathbf{S}, \mathbf{w}^{\text{prev}}$)    ▷ Compute client weights
4:     $\boldsymbol{\alpha} \leftarrow \boldsymbol{\alpha} / \sum_{i=1}^{m} \alpha_i$    ▷ Normalize the weights
5:     $\boldsymbol{\alpha} \leftarrow$ Softmax($\boldsymbol{\alpha}$)    ▷ Apply softmax
6:     for $k \in \mathbf{w}^{\text{global}}$ do
7:         $\mathbf{u} \leftarrow \mathbf{0}$
8:         for $i \leftarrow 1$ to $m$ do
9:             $\mathbf{u} \leftarrow \mathbf{u} + \alpha_i \mathbf{w}_i[k]$    ▷ Weighted aggregation of client updates
10:        end for
11:        $\mathbf{w}^{\text{global}}[k] \leftarrow \mathbf{u}$    ▷ Update global model
12:    end for
13:    $\mathbf{w}^{\text{client}} \leftarrow \mathbf{w}^{\text{global}}$    ▷ Update client models
14: end procedure
```

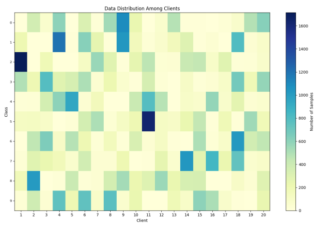
The overall FL algorithm incorporating the client similarity information is presented in Algorithm 1.

IV. EXPERIMENTS

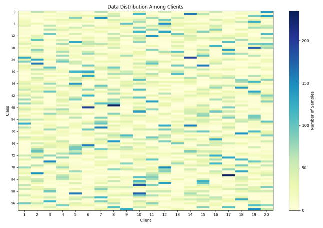
A. Experimental Setup

We conduct experiments on datasets, CIFAR-10 and CIFAR-100 [34] both of which are widely recognized as benchmark datasets in the context of heterogeneous FL. To ensure a rigorous evaluation, we utilized the ResNet-18 architecture for both datasets. ResNet-18, introduced by He et al. [35], is a well-established convolutional neural network (CNN) known for its residual learning framework, which effectively mitigates the vanishing gradient problem in deep networks. This architecture, consisting of 18 layers structured around residual blocks, has demonstrated robust performance in image classification tasks and serves as a consistent baseline in FL studies. While our primary focus is on introducing and validating a novel aggregation approach, ResNet-18 provides a reliable foundation for assessing the performance of our method across diverse experimental settings. We compare the proposed SimProx approach with the state-of-the-art method FedAvg, along with the related methods SimAgg [28] and FedProx [26].

The proposed method is implemented using Pytorch [36] and executed on a GPU-enabled cloud environment. The SGD optimizer is used with the learning rate set to 0.01 and weight decay to 0.001. The batch size is set to 64. The number of local epochs is set to 20 epochs for all FL approaches. For our approach, the optimal value of the lambda parameter was 0.7. For CIFAR-10 and CIFAR-100, the experiments were conducted 10 times, starting with 10 rounds, then repeated with 20 rounds and so on, up to 100 rounds.



(a) CIFAR-10 Data Distribution



(b) CIFAR-100 Data Distribution

FIGURE 2: Data Distribution Across 20 Clients for CIFAR-10 and CIFAR-100.

For collaborator selection, we use a subset of the available collaborators (30%) in each round. To accommodate system heterogeneity, where collaborator contributions may vary unpredictably, we simulate a random selection process in each round. To ensure that all collaborators are engaged uniformly over time, we utilize a sliding window mechanism over the randomized collaborator index. This method ensures that, after all collaborators have participated, a new randomized order is generated to enhance learning efficiency. The sliding window approach is favored over random selection to guarantee consistent participation from all collaborators. The sliding-window size is set to 30% of the total number of collaborators within each partition. This approach guarantees that each client contributes to the model's updates over time, while also introducing an element of randomness to prevent overfitting to any particular client order.

B. Experimental Results on CIFAR

The CIFAR-10 dataset contains 60,000 color images of size 32x32, divided into 10 mutually exclusive classes with 6,000 images per class. It includes 50,000 training images and 10,000 test images. The CIFAR-100 dataset consists of 100 classes, each containing 600 images, with 500 training images and 100 testing images per class. Figure 2 shows the samples of CIFAR10 and CIFAR100 distributed in the default settings across 20 clients participating in the FL process. Like previous studies [37], [38], we use Dirichlet distribution to generate the non-IID data partition among parties. Specifically, we sample $p_k \sim \text{Dir}_N(\beta)$ and allocate a $p_{k,j}$ proportion of the instances of class k to party j, where $\text{Dir}(\beta)$ is the Dirichlet distribution with a concentration parameter β (0.5 by default). With the above partitioning strategy, each party can have relatively few (even no) data samples in some classes. We set the number of parties to 20 where 6 clients are picked randomly to participate in the learning process in each round to simulate a real world scenario.

Table 2 displays the classification accuracy results on CIFAR-10 over multiple communication rounds for four different methods: FedAvg, FedProx, SimAgg, and SimProx. SimProx consistently achieves the highest accuracy, across

TABLE 2: Average accuracy on CIFAR-10 and CIFAR-100 datasets across multiple rounds.

Round	Dataset	FedAvg	FedProx	SimAgg	SimProx
10	**CIFAR-10**	0.317	0.336	0.256	0.351
	CIFAR-100	0.103	0.097	0.056	0.100
20	**CIFAR-10**	0.324	0.404	0.290	0.422
	CIFAR-100	0.106	0.134	0.078	0.136
30	**CIFAR-10**	0.366	0.438	0.334	0.450
	CIFAR-100	0.164	0.150	0.101	0.153
40	**CIFAR-10**	0.410	0.449	0.364	0.492
	CIFAR-100	0.193	0.176	0.122	0.177
50	**CIFAR-10**	0.419	0.503	0.356	0.449
	CIFAR-100	0.149	0.181	0.095	0.180
60	**CIFAR-10**	0.407	0.450	0.383	0.513
	CIFAR-100	0.133	0.182	0.123	0.185
70	**CIFAR-10**	0.446	0.530	0.403	0.528
	CIFAR-100	0.166	0.176	0.109	0.211
80	**CIFAR-10**	0.457	0.561	0.384	0.565
	CIFAR-100	0.190	0.196	0.112	0.227
90	**CIFAR-10**	0.491	0.559	0.378	**0.628**
	CIFAR-100	0.188	0.197	0.116	0.244
100	**CIFAR-10**	0.476	0.609	0.429	0.614
	CIFAR-100	0.195	0.198	0.058	**0.247**

all communication rounds compared to the other methods. The curve indicates that SimProx is particularly effective in handling the data, which improves convergence speed and model robustness. Both SimProx and FedProx show steady improvement, outperforming FedAvg and SimAgg and maintaining the lead throughout the evaluation process. However, SimProx shows approximately a 2% improvement in accuracy on CIFAR-10 compared to FedProx, reaching a peak accuracy of 0.628 with 90 rounds, whereas FedProx achieves its best accuracy of 0.609 with 100 rounds.

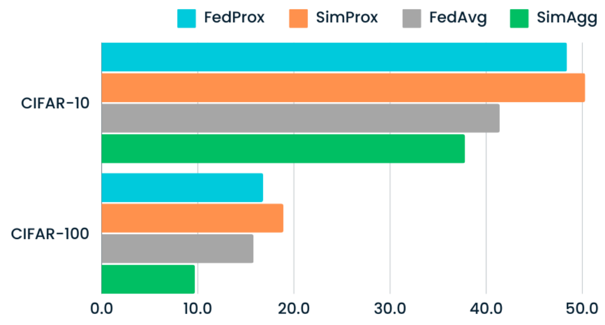
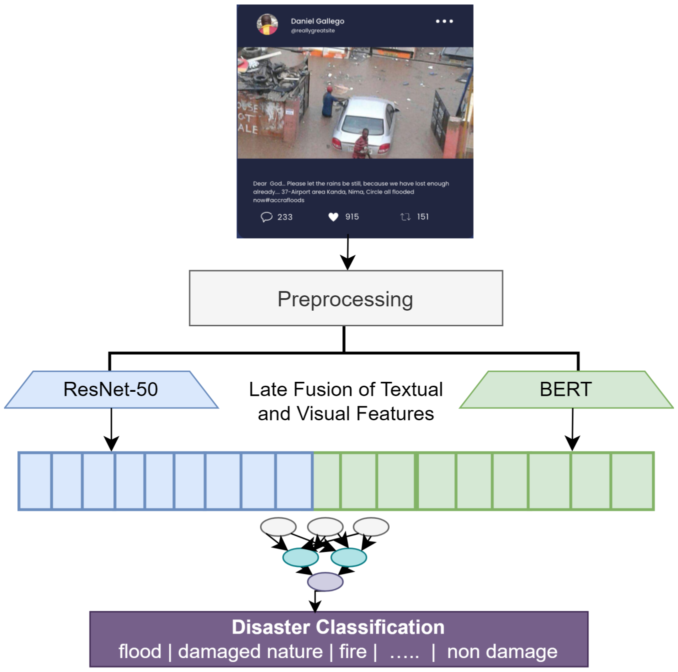


FIGURE 3: Average accuracy of all methods on CIFAR-10 and CIFAR-100 across 100 rounds.


Dear God... Please let the rains be still, because we have lost enough already.... 37-Airport area Kanda, Nima, Circle all flooded now#accrafloods

Daniel Gallego
@reallygreatsite
233
915
151
Preprocessing
ResNet-50
Late Fusion of Textual and Visual Features
BERT
Disaster Classification
flood | damaged nature | fire | ..... | non damage

FIGURE 4: The multimodal deep learning model used for disaster event classification.

Table 2 also demonstrates how SimProx performs better than the other methods on CIFAR-100 under the same non-IID settings, achieving the highest classification accuracy of 0.247 overall with 100 rounds. The results of the SimProx aggregation approach show superior and consistent performance starting from 60 rounds onward, earlier than the other methods. With a $5-6\%$ accuracy gain over FedAvg and FedProx, SimProx proves its reliability and effectiveness even with the more complex CIFAR-100 dataset.

Figure 3 presents the average accuracy scores obtained from ten test experiments, i.e., 10 to 100 rounds, conducted for each aggregation approach on CIFAR-10 and CIFAR-100. SimProx outperforms all the other methods, achieving the highest accuracy. These accuracy results suggest that SimProx offers a robust and efficient aggregation approach to FL.

V. PERFORMANCE ON REAL-WORLD MULTIMODAL DATA

To assess the performance of the proposed aggregation method on different type of data, we conducted a comprehensive evaluation on the MEDIC dataset [39], a publicly available dataset comprising image-tweet pairs related to various disaster events. This dataset serves as a representative real-world multimodal data source, presenting the challenges inherent in social media data during critical events, such as class imbalance and noisy data.

The MEDIC dataset consists of 5,831 image-tweet pairs, divided into a training set of 5,247 samples and a test set of 584 samples. To mitigate the class imbalance issue, we employed a range of data augmentation techniques, including random horizontal flipping, color jittering, and random rotation, to enhance the diversity of the training data.

We configured the FL setup with 20 client models, similar to the CIFAR experimental setup, randomly selecting clients to participate in each training round. The performance of our proposed aggregation method was evaluated against the traditional FedAvg algorithm implemented in [40] on the MEDIC dataset, with results reported in terms of key metrics, including accuracy, precision, recall, and F1-score, averaged over 31 training rounds.

For feature extraction, we leveraged pre-trained models to capture both textual and visual information, as shown in Figure 4. Specifically, we utilized the BERT model [41] to extract meaningful features from the textual content of the tweets, and the ResNet50 [35] architecture to extract visual features from the corresponding images. The extracted features were then combined using a late fusion approach, where the outputs of the text and image models were concatenated to form a unified representation, enabling our model to capture the complementary information present in both modalities.

The experimental results, presented in Table 3, demonstrate the superiority of SimProx aggregation method com-

TABLE 3: Performance comparison on the MEDIC dataset.

Method	Accuracy	Precision	Recall	F1-score	Round
FedAvg [40]	85.1%	85.8%	85.1%	85.2%	31
SimProx	**93.0%**	**93.2%**	**93.0%**	**93.0%**	**11**

pared to FedAvg, emphasizing its effectiveness in leveraging the multimodal nature of the data and the collaborative nature of the FL environment. The improved classification performance, in terms of accuracy, precision, recall, and F1-score, highlights the advantages of our approach in addressing the challenges inherent in real-world multimodal data.

The findings of this study demonstrate the effectiveness of our proposed aggregation method in the context of real-world multimodal data, highlighting its potential to improve disaster event classification performance while addressing data privacy and scalability concerns.

VI. CONCLUSION

In this article, we introduced SimProx, a novel similarity-based aggregation method for federated deep learning that optimizes client weights using a combination of cosine and Gaussian similarity measures. Our approach addresses the challenges of data heterogeneity by effectively weighting client contributions based on computed similarities. Experimental results demonstrate that SimProx significantly improves the accuracy and robustness of the aggregated global model, outperforming traditional methods like FedAvg, particularly in non-IID settings and real-world multimodal datasets. The proposed method not only enhances model performance but also reduces training time, showcasing its potential for practical applications in decentralized environments. While SimProx offers substantial advantages, it also introduces increased computational complexity, which may pose challenges in large-scale FL systems to maintain scalability. Exploring additional similarity measures and addressing communication and computational heterogeneity are also promising directions for future research.

REFERENCES

[1] C. Zhang, Y. Xie, H. Bai, B. Yu, W. Li, and Y. Gao, "A survey on federated learning," *Knowledge-Based Systems*, vol. 216, p. 106775, 2021.

[2] M. Ye, X. Fang, B. Du, P. C. Yuen, and D. Tao, "Heterogeneous federated learning: State-of-the-art and research challenges," *ACM Computing Surveys*, vol. 56, no. 3, pp. 1–44, 2023.

[3] S. Baker and W. Xiang, "Artificial Intelligence of Things for Smarter Healthcare: A survey of advancements, challenges, and opportunities," *IEEE Communications surveys and tutorials/IEEE communications surveys and tutorials*, vol. 25, pp. 1261–1293, 1 2023.

[4] B. Liu, N. Lv, Y. Guo, and Y. Li, "Recent advances on federated learning: A systematic survey," *Neurocomputing*, p. 128019, 2024.

[5] 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.

[6] B. Luo, W. Xiao, S. Wang, J. Huang, and L. Tassiulas, "Tackling system and statistical heterogeneity for federated learning with adaptive client sampling," in *IEEE INFOCOM 2022-IEEE conference on computer communications*, pp. 1739–1748, IEEE, 2022.

[7] M. Mendieta, T. Yang, P. Wang, M. Lee, Z. Ding, and C. Chen, "Local learning matters: Rethinking data heterogeneity in federated learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8397–8406, 2022.

[8] S. Alam, L. Liu, M. Yan, and M. Zhang, "Fedrolex: Model-heterogeneous federated learning with rolling sub-model extraction," *Advances in neural information processing systems*, vol. 35, pp. 29677–29690, 2022.

[9] Z. Zhu, J. Hong, and J. Zhou, "Data-free knowledge distillation for heterogeneous federated learning," in *International conference on machine learning*, pp. 12878–12889, PMLR, 2021.

[10] T. Lin, L. Kong, S. U. Stich, and M. Jaggi, "Ensemble distillation for robust model fusion in federated learning," *Advances in neural information processing systems*, vol. 33, pp. 2351–2363, 2020.

[11] A. Fallah, A. Mokhtari, and A. Ozdaglar, "Personalized federated learning: A meta-learning approach," *arXiv preprint arXiv:2002.07948*, 2020.

[12] E. Diao, J. Ding, and V. Tarokh, "Heterofl: Computation and communication efficient federated learning for heterogeneous clients," *arXiv preprint arXiv:2010.01264*, 2020.

[13] C. Xu, Y. Qu, Y. Xiang, and L. Gao, "Asynchronous federated learning on heterogeneous devices: A survey," *Computer Science Review*, vol. 50, p. 100595, 2023.

[14] J. Hong, H. Wang, Z. Wang, and J. Zhou, "Efficient split-mix federated learning for on-demand and in-situ customization," *arXiv preprint arXiv:2203.09747*, 2022.

[15] J. Yoon, G. Park, W. Jeong, and S. J. Hwang, "Bitwidth heterogeneous federated learning with progressive weight dequantization," in *International Conference on Machine Learning*, pp. 25552–25565, PMLR, 2022.

[16] C. Zhou, H. Tian, H. Zhang, J. Zhang, M. Dong, and J. Jia, "Tea-fed: time-efficient asynchronous federated learning for edge computing," in *Proceedings of the 18th ACM International Conference on Computing Frontiers*, pp. 30–37, 2021.

[17] Y. Chen, X. Sun, and Y. Jin, "Communication-efficient federated deep learning with layerwise asynchronous model update and temporally weighted aggregation," *IEEE transactions on neural networks and learning systems*, vol. 31, no. 10, pp. 4229–4238, 2019.

[18] L. Qu, Y. Zhou, P. P. Liang, Y. Xia, F. Wang, E. Adeli, L. Fei-Fei, and D. Rubin, "Rethinking architecture design for tackling data heterogeneity in federated learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10061–10071, 2022.

[19] Z. Qu, X. Li, R. Duan, Y. Liu, B. Tang, and Z. Lu, "Generalized federated learning via sharpness aware minimization," in *International conference on machine learning*, pp. 18250–18280, PMLR, 2022.

[20] Z. Tang, Y. Zhang, S. Shi, X. He, B. Han, and X. Chu, "Virtual homogeneity learning: Defending against data heterogeneity in federated learning," in *International Conference on Machine Learning*, pp. 21111–21132, PMLR, 2022.

[21] A. Nilsson, S. Smith, G. Ulm, E. Gustavsson, and M. Jirstrand, "A performance evaluation of federated learning algorithms," in *Proceedings of the second workshop on distributed infrastructures for deep learning*, pp. 1–8, 2018.

[22] H. Zhu, J. Xu, S. Liu, and Y. Jin, "Federated learning on non-iid data: A survey," *Neurocomputing*, vol. 465, pp. 371–390, 2021.

[23] 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.

[24] Y. Tan, Y. Liu, G. Long, J. Jiang, Q. Lu, and C. Zhang, "Federated learning on non-iid graphs via structural knowledge sharing," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 37, pp. 9953–9961, 2023.

[25] 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*, pp. 1273–1282, PMLR, 2017.

[26] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," *arXiv preprint arXiv:1812.06127*, 2018.

[27] D. A. E. Acar, Y. Zhao, R. Matas, M. Mattina, P. Whatmough, and V. Saligrama, "Federated learning based on dynamic regularization," *arXiv preprint arXiv:2111.04263*, 2021.

[28] M. I. Khan, M. Jafaritadi, E. Alhoniemi, E. Kontio, and S. A. Khan, "Adaptive weight aggregation in federated learning for brain tumor segmentation," in *International MICCAI Brainlesion Workshop*, pp. 455–469, Springer, 2021.

[29] 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.

[30] Q. Li, B. He, and D. Song, “Model-contrastive federated learning,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10713–10722, 2021.

[31] Z. Chen, S. Yu, F. Chen, F. Wang, X. Liu, and R. H. Deng, “Lightweight privacy-preserving cross-cluster federated learning with heterogeneous data,” *IEEE Transactions on Information Forensics and Security*, 2024.

[32] Y. Yan, X. Tong, and S. Wang, “Clustered federated learning in heterogeneous environment,” *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

[33] C. Briggs, Z. Fan, and P. Andras, “Federated learning with hierarchical clustering of local updates to improve training on non-iid data,” in *2020 international joint conference on neural networks (IJCNN)*, pp. 1–9, IEEE, 2020.

[34] K. Alex, “Learning multiple layers of features from tiny images,” *https://www. cs. toronto. edu/kriz/learning-features-2009-TR. pdf*, 2009.

[35] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[36] S. Imambi, K. B. Prakash, and G. Kanagachidambaresan, “Pytorch,” *Programming with TensorFlow: solution for edge computing applications*, pp. 87–104, 2021.

[37] M. Yurochkin, M. Agarwal, S. Ghosh, K. Greenewald, N. Hoang, and Y. Khazaeni, “Bayesian nonparametric federated learning of neural networks,” in *International conference on machine learning*, pp. 7252–7261, PMLR, 2019.

[38] H. Wang, M. Yurochkin, Y. Sun, D. Papailiopoulos, and Y. Khazaeni, “Federated learning with matched averaging,” *arXiv preprint arXiv:2002.06440*, 2020.

[39] F. Alam, T. Alam, M. A. Hasan, A. Hasnat, M. Imran, and F. Ofli, “Medic: a multi-task learning dataset for disaster image classification,” *Neural Computing and Applications*, vol. 35, no. 3, pp. 2609–2632, 2023.

[40] A. El-Niss, A. Alzu’Bi, and A. Abuarqoub, “Multimodal fusion for disaster event classification on social media: A deep federated learning approach,” in *Proceedings of the 7th International Conference on Future Networks and Distributed Systems*, pp. 758–763, 2023.

[41] A. Vaswani, “Attention is all you need,” *arXiv preprint arXiv:1706.03762*, 2017.