Subjective Logic-based Decentralized Federated Learning for Non-IID Data

Agnideven Palanisamy Sundar
Department of Computer and Information Science,
Indiana University-Purdue University Indianapolis
Indianapolis, Indiana, USA
agpalan@iu.edu

Feng Li
Department of Computer and Information Technology,
Indiana University-Purdue University Indianapolis
Indianapolis, Indiana, USA
fengli@iupui.edu

Xukai Zou
Department of Computer and Information Science,
Indiana University-Purdue University Indianapolis
Indianapolis, Indiana, USA
xzou@iupui.edu

Tianchong Gao
School of Cyber Science and Engineering,
Southeast University
Frontiers Science Center for Mobile Information
Communication and Security, Southeast University
Nanjing, Jiangsu, China
tgao@seu.edu.cn

ABSTRACT

Existing Federated Learning (FL) methods are highly influenced by the training data distribution. In the single global model FL systems, users with highly non-IID data do not improve the global model, and neither does the global model work well on their local data distribution. Even with the clustering-based FL approaches, not all participants get clustered adequately enough for the models to fulfill their local demands. In this work, we design a modified subjective logic-based FL system utilizing the distribution-based similarity among users. Each participant has complete control over their own aggregated model, with handpicked contributions from other participants. The existing clustered model only satisfies a subset of clients, while our individual aggregated models satisfy all the clients. We design a decentralized FL approach, which functions without a trusted central server; the communication and computation overhead is distributed among the clients. We also develop a layer-wise secret-sharing scheme to amplify privacy. We experimentally show that our approach improves the performance of each participant's aggregated model on their local distribution over the existing single global model and clustering-based approach.

CCS CONCEPTS

• **Security and privacy** → *Distributed systems security.*

KEYWORDS

Federated Learning, Subjective Logic, Distributed Systems, Generative Adversarial Networks, Non-IID Data

ARES 2024, July 30–August 02, 2024, Vienna, Austria

ACM ISBN 979-8-4007-1718-5/24/07
https://doi.org/10.1145/3664476.3664517

ACM Reference Format:
Agnideven Palanisamy Sundar, Feng Li, Xukai Zou, and Tianchong Gao. 2024. Subjective Logic-based Decentralized Federated Learning for Non-IID Data. In *The 19th International Conference on Availability, Reliability and Security (ARES 2024), July 30–August 02, 2024, Vienna, Austria.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3664476.3664517

1 INTRODUCTION

Since its inception, Federated Learning (FL) has garnered immense attention from the research community, corporations, and individuals. The ability to access and contribute to high-quality models without the need for massive datasets has made FL popular across domains. Though FL exhibits exceptional benefits theoretically, those benefits are unattainable in most practical cases.

In the context of FL, Independent and Identically Distributed (IID) data implies FL system assumes that each class in the local training data has the same number of data records across all the participating clients. This assumption is not practical in real-world FL setups since each client would have different requirements and data collection strategies. On the other hand, non-IID data assumption does not require the clients to have similar local data distribution.

Some non-IID FL protocols also use the single global model approach [29], where all the clients' local models are aggregated to produce one global model. The requirement of a non-IID client differs significantly from that of an IID client. The nature of the distribution of the client's training data is a testament to the client's evaluation requirements. If a client's local data is extremely non-IID, the client might expect his model to perform well on similar data distribution. A single global model approach cannot satisfy all the non-IID clients with the same model.

The clustering-based FL systems are a subset of FL protocols that work well for non-IID data. Clustering methods group clients into different clusters based on the similarities in their model features [25, 27]. Then each cluster forms its own cluster model. These clusters are a good alternative for single global models, but still, they don't completely satisfy clients with highly non-IID data. Clients not closely related to other clients are grouped into an irrelevant cluster or left out as outliers. Both these situations are equally

unfavorable. Moreover, two unrelated cluster models will have an abysmal performance on each others' distribution, which is undesirable.

In the single global model approach and clustering approach, the participants are treated as part of a group, not as individual clients. Every client joining an FL system enters with an expectation to improve the performance of his model. He assumes that all the other models in the network would help enhance its performance, but, in many cases, it could be the contrary. If each user could build their own FL model based on their requirements, then all the clients in the system would be satisfied with the protocol rather than just a subset of clients.

In our work, we design a decentralized FL framework using the principles of Subjective Logic. Each client will directly evaluate the other clients and use the information to determine the models that will be a part of their Personal Aggregated Model. We make the following contribution:

(1) We design a completely decentralized FL framework, where each client assembles their own Personal Aggregated Model suitable for their data distribution.
(2) We develop a GAN-based similarity metric, which allows the clients to check their similarities with other clients in the network without leaking their original training data.
(3) We build a Subjective Logic-based evaluation method that allows users to shortlist clients for the aggregation process.
(4) We design a layer-wise secret sharing scheme that allows clients to share models without breaching privacy.
(5) We experimentally evaluate the performance of our approach against Single Global Model FL, Clustered Model FL, and SOTA approaches under both IID and non-IID conditions.

This paper is organized as follows: In Sec. 2, we discuss the prerequisite information regarding Federated Learning, Subjective Logic, and GAN. Then, we look at some of the closely related works in Sec. 3. Followed by that, in Sec. 4, we look at a high-level overview of our approach and discuss the threat model and assumptions. In Sec. 5, the methodology, we explore the various steps involved in our decentralized FL approach in detail. It is followed by Sec. 6, which is the experiment section that consists of the parameter setup and evaluation of the approach in both IID and non-IID settings. We discuss the limitations and the future works we have in plans in Sec. 7. At last, we conclude paper in Sec. 8.

2 BACKGROUND

2.1 Federated Learning

Federated Learning is the process of fusing multiple local models into a combined model encompassing the properties of all the local models. FedAvg [22] was the first aggregation method introduced, and is still one of the most common aggregation methods used. This aggregation process is carried out for multiple rounds until the global model reaches the desired performance standard. Each participant in the system sends their model updates to the central server, where FedAvg is applied to get the current global model.

$$M^{t+1} = M^t + \frac{\eta}{N}\sum_{i=1}^{N}(C_i^{t+1} - M^t) \tag{1}$$

In the equation above, M^t is the current global model at iteration t, C_i^{t+1} is the model update of client i, N is the number of clients in the current round, and η is the global model learning rate. The global model M^t represents the combined model, which is expected to have the desired properties of the local models of all the N clients.

2.2 Data Distributions

Most of the existing classification models in machine learning are trained using data collected under different classes. Based on the distribution of the data in those classes, it can be categorized as either Independent and Identically Distributed(IID) or non-IID.

2.2.1 IID Data. In the context of FL, IID data implies that FL process assumes that each class in the local training data has the same number of data records across all the participating clients. This assumption is not practical in real-world FL setups, since each client would have different requirements and data collection strategies. Though this is impractical, this assumption makes it easier to build global models with good performance with relative ease.

2.2.2 Non-IID. On the other hand, non-IID data assumption does not require the clients to have similar local data distribution. It is also possible for users to have data records only for some of the classes involved in the FL system. Such variations in the local data distribution creates differences in the local models trained by different clients. These differences make FL under non-IID assumption much more tedious than IID assumption.

2.3 Subjective Logic

Subjective Logic (SL) is a probabilistic logic predominantly applied in trust and Bayesian networks [15, 16]. On a high level, SL is used to form the Subjective Trust Network, which helps each node in the network to establish a trust opinion of other nodes, which may or may not be in direct contact with this node. It helps determine the credibility of a node for a given proposition. There are many variations of SL, but the initial binomial version consists of three factors for any given proposition x: 1) belief mass (b_x) is the extent to which x is calculated to be true, 2) disbelief mass (d_x) is the extent to which x is considered false, and 3) uncertainty mass (u_x), is the level of epistemic uncertainty involved in the calculations. The sum $b_x + d_x + u_x = 1$. These factors are calculated by each node for its neighbors and are updated after every interaction. The values of b_x, d_x, and u_x are evaluated based on the number of positive and negative interaction evidence among the nodes. The uncertainty in the calculation drops as the number of interactions increase.

2.4 Generative Adversarial Networks

A Generative Adversarial Network is a subclass of Machine Learning whose objective is to generate data similar to the source data used for training [8]. GANs are most commonly used in the computer vision domain to generate photo-realistic synthetic images. GAN-generated data records help extend the overall dataset in situations where the size of the local data is small. GAN needs to be well-trained before extending the local data, which can be time-consuming. But, the Generator can produce functional synthetic data, with minimal source data properties, from the early stages

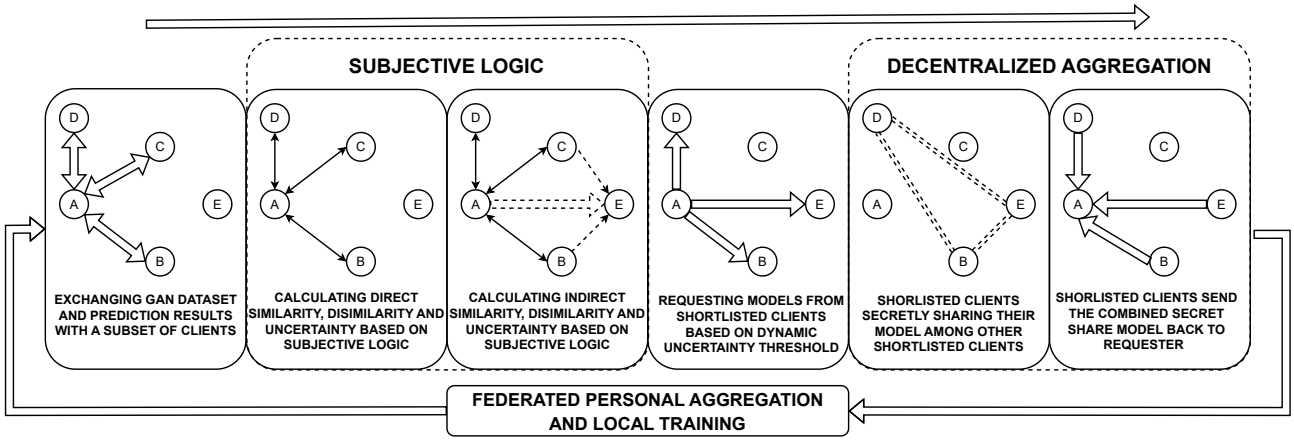


Figure 1: An overview of all the major steps in our approach. These steps are followed by each of the clients in the FL process for each round to obtain their own Personal Aggregation Model, which best suits their local requirements.

of training. We only require functional synthetic data for our purposes and hence won't have high time complexity. Moreover, under trained data records do not leak any information about the source data.

3 RELATED WORK

3.1 Federated Learning with Non-IID

Many recent works have discussed the distribution issues in Federated Leaning and have devised methods that work under certain conditions [24, 33, 35, 37]. In [38], Zhao et al. state that non-IID data causes divergence in the neural network layers. They create a small subset of data shared among all the clients to reduce divergence. In [34], the authors develop a framework called Favor that can select the appropriate clients for each round to counterbalance the bias induced by non-IID data. In [17], Li et al. mitigate model feature shifts by local batch normalization before averaging. In [2], the authors adopt a local update similarity-based clustering approach, with each cluster having its global model. In [4], Chen et al. design a system where the central model is updated asynchronously and the edge devices perform continuous learning by streaming local data. Many of the existing works rely on some form of clustering approach or only produce a single global model to accommodate all the client's needs [3, 13, 23]. On the other hand, our work produces individual global models that best suits each of the client's requirements.

3.2 Decentralized Federated Learning

Most of the existing decentralized FL methods mainly focus on the IID dataset. In [11], a segment-level decentralized FL approach is designed by utilizing the node-to-node bandwidth. In [10], the authors compare gossip learning with federated learning and propose gossip learning can be used as a decentralized FL alternative. They claim that gossip learning works better than FL in the IID setting. In [28], Roy et al. build a torrent-like decentralized FL method, where versions of models are maintained by the clients in a peer-to-peer network. In [26], a decentralized FL based on Interplanary File System (IPFS) is developed. Clients can be connected to the IPFS network and initiate or join the training process. There is no information on whether it applies to non-IID settings. A blockchain-based decentralized FL is designed in [18]. Li et al. use the blockchain with committee consensus for storing the global model and exchanging the local models. Our approach differs from all this because of how each client can decide the contributors of its PAM. Many other approaches use clients to determine the functioning of the common global model.

4 OVERVIEW

This section discusses an overview of our approach. Fig. 1 gives a gist of all the steps involved in our approach. Here, we analyze the process from the perspective of Client A. All the clients follow the same protocol as Client A to get their Personal Aggregation Model.

4.1 Steps Involved: High-Level

(1) Firstly, Client A randomly selects some other clients (B, C, and D in this case) to work with on that round and exchanges its GAN dataset with them. In return, it receives the prediction results of the other clients on its GAN data.
(2) Client A applies Subjective Logic to the clients that are directly connected through the previous step to get the similarity, dissimilarity, and uncertainty scores.
(3) Subjective Logic is then applied again to get the scores for clients that are not directly connected in this round (E). These indirect connections are only obtained through information shared by the direct connections formed in that round.
(4) Then, Client A calculates a dynamic uncertainty threshold that is used to shortlist the clients who would be suitable for the aggregation step and a request is sent. In the toy example, it is clients B, D, and E.
(5) To enhance security, the other clients first share their model's layer-wise secret shares among themselves before sending them back to Client A.
(6) Once Client A receives all the relevant shares for aggregation. It combines all the shares with its own local model and get

the Personal Aggregated Model. Further local training is conducted on this model. These steps are repeated in all rounds.

The motivation behind our approach is to allow each client the flexibility to build their own FL model. This ensures that the model's performance suits its local data requirements well.

5 METHODOLOGY

5.1 Initial Local Training and GAN Dataset Generation

For ease of understanding, let us look at all the functions from the perspective of a single client, the Representative Client (RepClient). All the clients will follow the same protocol as the RepClient, throughout the FL process. At the beginning of the FL process, the RepClient trains its local model only based on its local dataset. The hyperparameters and the initialization weights needed for the training are part of the FL protocol.

Simultaneously with the local model training, the RepClient also trains a weak Generative Adversarial Network to produce synthetic data records similar to its local data. The GAN only needs to be trained for a few epochs; the actual number of training epochs would vary depending on the local dataset. The aim here is only to generate data records, which are better than random vector data in representing the features of the local data. Given that the GAN is under-trained, there will be negligible information leakage about the source data. Privacy-preserving GAN variants [19, 32] can also be used to enhance privacy. In each round where GAN is trained, a different seed input is used for GAN to improve privacy further.

5.2 Similarity Metrics

5.2.1 ***Class Prediction Count.*** Once the GAN data records are generated, the RepClient tests those records against its local model and estimates the number of records predicted for each class. We call this the Class Prediction Count (CPC, denoted by Δ). For instance, if there are 5 total classes in the FL training, and the GAN dataset has 1000 records, then the $\Delta^{RepClient}_{RepClient}$ could be [151, 0, 356, 469, 24]. Each value denotes the number of records classified into its corresponding class. Δ^A_B signifies the CPC for Client A model's performance for Client B's GAN dataset

5.2.2 ***Prediction Similarity Score.*** After calculating its own CPC, the RepClient randomly selects a small subset of other clients, known as Direct Subset (DS), and shares the GAN dataset with them. The DS clients also send their GAN dataset back to the RepClient. The RepClient calculates CPC individually for the received datasets. The CPC values are sent back to the respective clients, along with the CPC value of its own GAN dataset. Similarly, the DS clients send back their CPCs for RepClient's GAN dataset and their own GAN dataset.

Based on the CPCs received, the RepClient can calculate the similarity between its local model and other clients' models. We call this the Prediction Similarity Score (PSS, denoted by λ); it is the Cosine Similarity between the RepClient's CPC and each of the DS clients' CPC. Client A can calculate the PSS score for Client B given as λ^A_B by:

$$\lambda^A_B = \frac{1}{2}\left[\frac{\Delta^A_A \cdot \Delta^B_A}{||\Delta^A_A|| * ||\Delta^B_A||} + \frac{\Delta^A_B \cdot \Delta^B_B}{||\Delta^A_B|| * ||\Delta^B_B||}\right] \tag{2}$$

In the above equation, we take the average of the cosine similarities of both the client's CPCs on both the client's GAN datasets. We further fine-tune the PSS value by taking the performance average on both the GAN datasets. By taking the average of the two, we better capture the true similarity between the models.

5.3 Subjective Logic

Though we have the PSS with all the users in the DS, it does not precisely reflect the exact similarity between the models. This deficiency can be attributed to multiple factors, including the size of the GAN dataset and the variance introduced by the training period. Once the necessary PSSs are calculated, the RepClient applies a variation of the SL in two phases. The SL metric χ between two clients consists of three parts, χ^A_B is the set containing $\{s^A_B; d^A_B; u^A_B\}$, the SL value calculated by Client A for Client B.

5.3.1 ***Phase 1- Direct Information.*** In each round, the RepClient forms a direct link with the clients in the DS.

$$s^A_B = \frac{\alpha}{\alpha + \beta + 2} \tag{3}$$

$$d^A_B = \frac{\beta}{\alpha + \beta + 2} \tag{4}$$

$$u^A_B = \frac{2}{\alpha + \beta + 2} \tag{5}$$

$$s^A_B + d^A_B + u^A_B = 1 \tag{6}$$

Here s^A_B, d^A_B, and u^A_B denote the SL-based similarity, dissimilarity, and uncertainty, respectively, calculated by Client A for Client B. Initially, both α and β are equal to 1. Greater s^A_B means that A and B are similar. Greater d^A_B mean that A and B are highly different from each other. Greater u^A_B means that further investigation is needed to determine the actual closeness of A and B. We increment both α and β values, simultaneously, for each direct interaction, with $\alpha = \alpha + \lambda^A_B$ and $\beta = \beta + [1 - \lambda^A_B]$. All the above SL calculations are only used for the DS clients in each round.

5.3.2 ***Phase 2- Indirect Information.*** The number of direct links that can be formed efficiently in each round is minimal. If we were to consider an FL system with thousands of participants, developing direct links among all the clients would be computationally inefficient. This inadequacy can be overcome by sharing the SL information among the clients.

After phase 1, the RepClient and the DS clients exchange their calculated SL scores. Upon receiving the scores, the RepClient applies two rules to integrate this Indirect information into its SL calculations.

Discounting. If Client A has a direct connection to Client B, and Client B has SL scores for Client C, then the transitive similarity

between Client A and Client C, based on Client B, is calculated as:

$$\chi_C^{A\circ B} = \chi_B^A \odot \chi_C^B \tag{7}$$

$$s_C^{A\circ B} = s_B^A s_C^B \tag{8}$$

$$d_C^{A\circ B} = s_B^A d_C^B \tag{9}$$

$$u_C^{A\circ B} = d_B^A + u_B^A + s_B^A u_C^B \tag{10}$$

Consensus. Multiple clients in the DS can have SL scores for C in the same round. Even Client A might have its own SL score for Client C. In such cases, all the scores can be combined using consensus rule.

$$\chi_C^{A:B} = \chi_C^A \oplus \chi_C^B \tag{11}$$

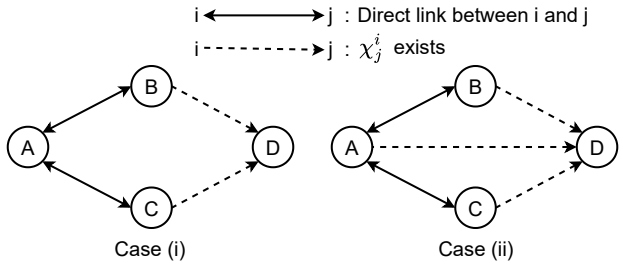
$$s_C^{A:B} = \left(s_C^A u_C^B + s_C^B u_C^A\right)/K \tag{12}$$

$$d_C^{A:B} = \left(d_C^A u_C^B + d_C^B u_C^A\right)/K \tag{13}$$

$$u_C^{A:B} = \left(u_C^A u_C^B\right)/K \tag{14}$$

$$\text{where} \quad K = u_C^A + u_C^B - u_C^A u_C^B \tag{15}$$

Generally, the uncertainty associated with Indirect information is much higher than with direct connection.



For (i): $\chi_D^A = \chi_D^{[A\circ B]:[A\circ C]} = [\chi_B^A \odot \chi_D^B] \oplus [\chi_C^A \odot \chi_D^C]$

For (ii): $\chi_D^A = \chi_D^{[A]:[A\circ B]:[A\circ C]} = [\chi_D^A] \oplus [\chi_B^A \odot \chi_D^B] \oplus [\chi_C^A \odot \chi_D^C]$

Figure 2: Examples of combining SL scores using discounting and consensus rules.

Fig. 2 shows an example from the perspective of Client A. It has two cases showing how the χ_D^A needs to be updated in a round, depending on the existing SL scores and the direct links formed in that round.

5.4 Selecting Clients for Personal Aggregated Model

We use two factors to shortlist the clients (PAM_{list}) who will be a part of the Personal Aggregated Model (PAM).

5.4.1 ***Dynamic Uncertainty Threshold.*** As mentioned earlier, the uncertainty associated with each client varies depending on the number of Direct or Indirect contact it has had with the RepClient. If the result of the interaction in each round is consistently positive or consistently negative, then the uncertainty associated gradually decreases. But, if the result of the interaction fluctuates, then the uncertainty is maintained high. Even if the similarity of a client seems good, if the uncertainty is high, then the client does not get to contribute to the PAM. In such cases, an uncertainty threshold must be applied, above which the client is not selected. We create a Dynamic Uncertainty threshold (τ) measure to accommodate the varying uncertainty .

$$\tau = \begin{cases} avg\left(U_{list}\Big|_1^N\right) & \text{, if } avg(U_{list}) < med(U_{list}) \\ avg\left(U_{list}\Big|_1^{N/2}\right) & \text{, if } avg(U_{list}) \geq med(U_{list}) \end{cases} \tag{16}$$

Here U_{list} is the set of all the known SL uncertainty values estimated by a given client, arranged in ascending order with the smallest uncertainty score in position 1 and the largest in position N. N is the total number of uncertainty values in U_{list}. The function $avg(\cdot)$ return the average value of the input, and $med(\cdot)$ gives the median value.

The purpose of using an uncertainty threshold is to minimize the number of clients in the PAM_{list}, especially in the earlier rounds. The bigger the PAM_{list}, the greater the communication overhead among clients. By using 16, we ensure that the number of users contacted is always maintained to be lower than 50% of the clients in U_{list} until the overall uncertainty drops significantly.

5.4.2 ***Similarity-based Rescaling.*** Since similarity, dissimilarity, and uncertainty sum up to 1, a decrease in uncertainty means an increase in similarity or dissimilarity. The PAM should reflect the extent of similarity between the RepClient and the other clients in PAM_{list}. To achieve this, the RepClient rescales the models in PAM_{list} by multiplying each client's model weights with the their respective similarity scores, $\sigma_i = s_i^{RepClient}$, where $i \in PAM_{list}$. Such a rescaling ensures that the models with higher similarity contribute more substantially to the RepClient's PAM than the models with smaller similarity. If we used an approach where the dissimilar models are completely neglected from contributing to the Personal Aggregated Model, then PAM's performance on foreign data would be poor.

5.5 Decentralized Aggregation

Once the other client models are selected, and the respective scaling ranges are determined, the models need to be aggregated with the RepClient's local model. First, the RepClient now communicates its requirements to the selected clients.

5.5.1 ***Model Request Message.*** As a part of the model request, the RepClient sends each selected client a message consisting of two components. The first component is the similarity-based rescaling value σ corresponding to the client. The second component in the message is the list of the other clients who will also be contributing to RepClient's Personal Aggregated Model. Fig. 3(i) shows a toy example of the model requesting process. Client A sends a model request message to Clients B, C, D, and E.

5.5.2 ***Layer-wise Secret Shares.*** Suppose one client's model is directly shared with another client; the other client can execute a membership inference attack or a reconstruction attack [12]. To

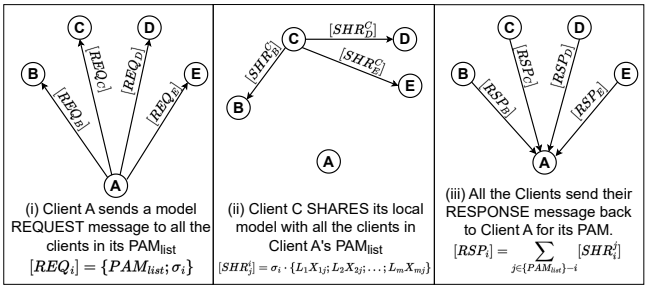


Figure 3: Communication steps involved in Decentralized Aggregation process.

avoid such privacy breaches, we split the client's model into multiple shares such that the original model cannot be revealed unless all the shares are combined. In essence, our approach is a modified version of [31], without the need to recover the original message as such. So, instead of directly sending the local model to the RepClient, a client splits the local model into multiple shares and sends it to all the clients in the RepClient's PAM_{list}.

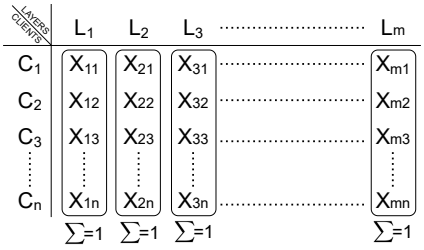


Figure 4: Layer-wise secret shares such that the sum of the shares sent to all the clients for each layer is equal to the original layer.

To execute this, we explore a layer-wise sharing approach. As a first step, all the clients rescale their local models using the σ they received. After rescaling, the model must be shared with the other clients in PAM_{list}. Fig. 4 shows the condition that needs to be satisfied by each share. Here n is the number of clients in the PAM_{list}, and m is the number of layers in the neural network model. $0 \leq X_{ij} < 1$ denotes the scalar weight that needs to be multiplied by the client's rescaled model vector weight in layer $i \in [1, m]$ to get that layer's share for client $j \in [1, n]$. The sum of all the scalar weights for each layer should add up to 1, $\sum_{j=1}^{n} X_{ij} = 1$. The values of the scalar weights are kept secret and never shared with other clients. Each client i sends a share to other clients consisting of:

$$[SHR_j^i] = \sigma_i \cdot \{L_1X_{1j}; L_2X_{2j}; \ldots; L_mX_{mj}\} \quad \text{where } j \in \{PAM_{list} - i\} \tag{17}$$

Fig. 3(ii) shows how Client C shares its local model between Clients B, D, and E.

5.5.3 ***Client Response Message.*** After each client shares their local models with other clients, they are left with multiple shares from all the other clients in PAM_{list}. Now, each client aggregates all the received shares and sends them back to the RepClient. The response message from each client i consists of:

$$[RSP_i] = \sum_{j \in \{PAM_{list} - i\}} [SHR_i^j] \tag{18}$$

Fig. 3(iii) gives an example of Clients B, C, D, and E, who are part of Client A's PAM_{list}, sending their response messages back to the requesting client, Client A.

5.5.4 ***Personal Model Aggregation.*** Once all the responses are received, the RepClient aggregates them based on their individual σ value. Since all the models have been rescaled, the aggregation should be divided by the rescaling factor:

$$PAM^{RepClient} = \frac{M_{loc}^{RepClient} + \sum_{i \in PAM_{list}} RSP_i}{1 + \sum_{i \in PAM_{list}} \sigma_i} \tag{19}$$

Here, $M_{loc}^{RepClient}$ denotes the RepClient's local model. It would be the PAM from the previous round unless it is the first round local model.

6 EXPERIMENTS

In this section, we evaluate the performance of our approach under different circumstances.

6.1 Setup and Baseline

6.1.1 ***Dataset.*** For our experiments, we use the CIFAR-10 image dataset, which consists of 50,000 training and 10,000 testing records. CIFAR-10 has the train and test data evenly split among its 10 classes, ranging from 'airplane', 'bird', 'cat' to 'truck'. We also use a subset of ImageNet32 dataset [5], which is a downsampled variant of ImageNet dataset with an image of size 32x32. We randomly select 40 classes which have more than 1000 data records in each class for our testing purposes.

Figure 5: Sample GAN images used for CPC calculation.

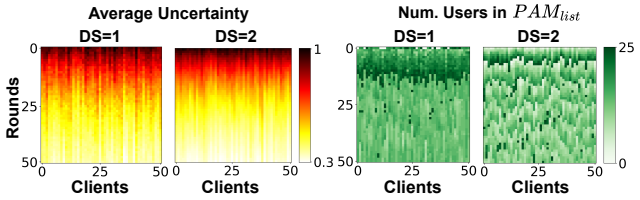


Figure 6: Average Uncertainty and number of clients in PAM_{list} for all clients for the first 50 rounds

6.1.2 ***Distribution.*** We split the CIFAR-10 data among 50 clients with data repetition under different distributions to check the effectiveness of our approach in an extreme situation. **Dist-0.** In this distribution, Each client gets 2k training records in 3 classes, 1.5k in 2 classes, 1k in 2 classes, and 5 in 3 classes. **Dist-1.** Every client has 2.5K training data records in 3 of the 10 classes and 5 in all other

classes. **Dist-2.** All the clients have 2K training records in 7 of the 10 classes and 5 in all other classes. **Dist-3.** This is a combination of all the other distribution methods. 8 clients have a distribution similar to Dist-0, 7 clients similar to Dist-1, and 7 clients similar to Dist-2. 8 clients have an even distribution, with 2k images in all training classes. The 20 remaining clients have a random distribution of data in all the classes, ranging from a minimum of 1 data record to a maximum of 4k records per class. In all the distributions, the classes that get more images are selected randomly for each client. Out of all the distributions, Dist-3 is the closest to real-world distribution and will be used in evaluation against other approaches. For the ImageNet32 distribution, we split the data among 100 clients based on Dirichlet distribution with parameters 0.6 and 0.3 as in [7].

6.1.3 ***Parameters.*** We run our experiments with a PyTorch code on Dell G5 laptops with Intel Core i7-10750H CPU, 16 GB of RAM, and NVIDIA GeForce GTX 1660 Ti GPU. We use ResNet18 architecture to train all the local models [9]. We use a vanilla GAN implementation for generating images for each of the clients [8]. The quality of the images is not expected to be good; Fig. 5 shows some of the samples of GAN images used in the CIFAR10 experiments. In each round, the client sends 1K GAN-generated images to the members in Direct Subset (DS).

The left two graphs of Fig. 6 shows the average uncertainty of the 50 clients in Dist-3 for the first 50 rounds of the FL process. We notice that the average uncertainty is much smoother when size of DS=2 and decreases faster than DS=1. The right two graphs in Fig. 6 represent the number of clients in the PAM_{list} for the same settings. We can notice the number of clients is always below 26. The fluctuations are more when DS=2, showing that the average and median uncertainty are close across rounds. Such closeness is possible if each client's U_{list} is consistent with a normal distribution. For this to be true, the majority of the uncertainties in the U_{list} need to be updated in each round; otherwise, the distribution would be skewed. We noticed that for both DS values, all the clients were directly or indirectly reached at least once by all other clients before round 4. We set DS=2 for all the experiments as it reaches enough clients each round to maintain a good distribution.

6.1.4 ***Performance Metrics.*** We use prediction accuracy of the individual clients as the metric to evaluate the performance of our approach. Since there are 50 clients for CIFAR10, we measure the average, minimum, and maximum accuracy across all clients. We use two different test dataset distributions.

Local-non-IID. It is non-IID, and the distribution is similar to the client's local distribution.

Global-IID. This is an IID distribution of all the test images in the CIFAR-10 test dataset. Such a distribution might not be necessary for practical situations, but it still helps us evaluate other users' contributions.

6.1.5 ***Baseline Comparison.*** To verify the validity of our claims, we test our method with CIFAR10 against two widely accepted Federated Learning approaches. Though our approach is decentralized and does not require a central server, we will compare it against established centralized approaches. We only use Dist-3 for comparison against other methods since it is the closest to a real-world scenario.

Single Global Model. Most existing FL systems still adopt a Single Global Model (SGM) framework. This method works well for IID data but is not always suitable for non-IID though it is widely used. We will test against the most known SGM protocol, the Federated Averaging method (FedAvg) [22].

Clustered Model. This technique acknowledges the influence of non-IID data on the global model. So, instead of using a single global model, the clients are separated into multiple clusters based on their features. Then all the client model's in each cluster combine to get a global model for the cluster. Many variations of clustering methods for FL have been developed recently [6, 25, 27, 36]. For our comparison, we choose one of the most effective techniques. To obtain the features needed for clustering, we find the pairwise cosine similarity between the output layers of all the client's local models. Then the HDBSCAN clustering method is used to group the clients into different clusters [21]. After clustering, each cluster forms its own Clustered Model (CM). The 50 clients in Dist-3 were grouped into 8 clusters.

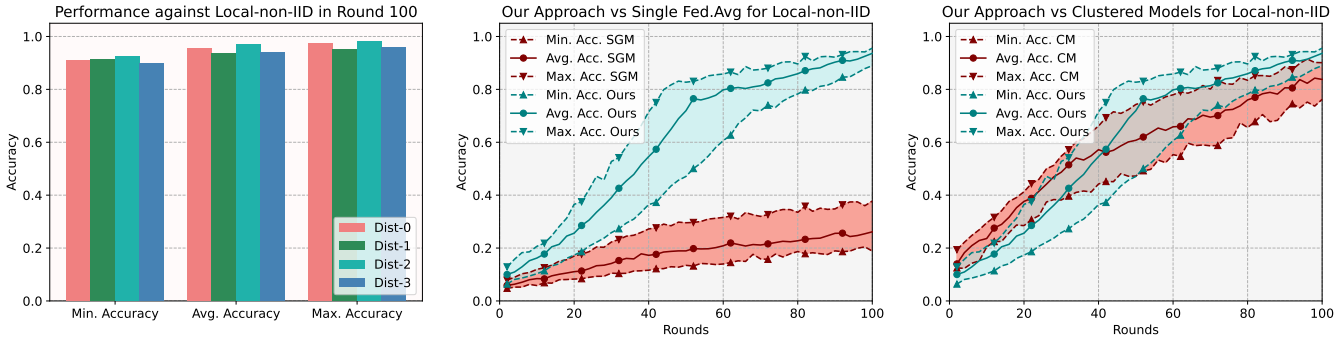
6.2 Evaluation on Local-non-IID Data

It is important for the client's Personal Aggregated Model to perform well in its local distribution. In this subsection, we test all the client's PAM's against their local distribution.

6.2.1 ***Impact of Distribution Extent.*** Here, we test the four distributions and check the minimum, maximum, and average accuracy of all 50 clients. Fig. 7a shows the performance of our SL-FL approach under different distributions at training round 100. From the figure, all the distributions have a minimum accuracy of over 90%. Moreover, the average accuracy of all the clients for all the distributions is over 94%. Given that the distribution of all the clients in Dist-1 is highly skewed, our approach still manages to extract an exceptional performance on their Local-non-IID data.

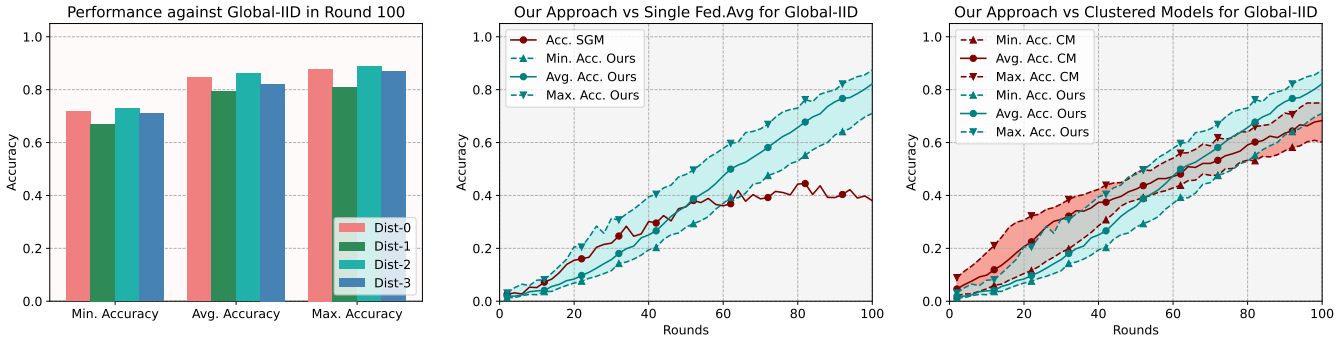
6.2.2 ***Performance Against SGM.*** When evaluating our approach against Single Global Model for Dist-3, we noticed that the performance of the SGM was poor for almost all the client's Local-non-IID. The poor performance is because of the contrast in the weights of the local models of each client. Since the training data distribution of all the clients is highly varied, their model weights may cancel each other out in the SGM approach. Fig. 7b shows that the average accuracy of the SGM on Local-non-IID is only close to 25%, even after 100 rounds, whereas our approach has an average accuracy of over 94%.

6.2.3 ***Performance Against CM.*** The Local-non-IID of each client was evaluated using their corresponding clusters. Since the global model executes the clustering, it has a global view of all the clients, allowing for a faster matching among similar clients. But the CM's advantage over our approach is present only for the first few rounds. As seen in Fig. 7c, after round 40, our method consistently maintains about 10% higher average accuracy than CM. Our approach, which is far more fine-tuned with the selection and rescaling process, successfully manages to keep the essence



(a) Different Distributions on Local -non-IID **(b) Our Approach vs. Single GM on Local -non-IID** **(c) Our Approach vs. Clustered on Local-non-IID**

Figure 7: Subfigure (a) shows the performance of our approach under different distributions on CIFAR10 Local-non-IID test data. Subfigures (b) and (c) show the performance comparison of our approach against the Single Global Model approach and the Clustering-based approach on CIFAR10 Local-non-IID test data.



(a) Different Distributions on Global-IID **(b) Our Approach vs. Single GM on Global-IID** **(c) Our Approach vs. Clustered on Global-IID**

Figure 8: Subfigure (a) shows the performance of our approach under different distributions on CIFAR10 Global-IID test data. Subfigures (b) and (c) show the performance comparison of our approach against the Single Global Model approach and the Clustering-based approach on CIFAR10 Global-IID test data.

of the local distribution far better than the CM, though it has the advantage of being centralized.

6.3 Evaluation on Global-IID Data

The Global IID data has an equal number of data records for all the classes in the test dataset. Though rarely required, it is better to have a good overall performance.

6.3.1 ***Impact of Distribution Extent.*** Regarding the performance on Global-IID data, the overall distribution of all the participants play a crucial role. As shown in Fig. 8a, overall performance has dropped for all the distributions. At round 100, the minimum accuracy for all the distributions is less than 75%. But, the lowest average accuracy is almost 80%. This shows that there are many fluctuations in the client's performance when it comes to Global-IID data. The lowest average accuracy is almost 80% for Dist-1, which is great considering that it is an extreme non-IID distribution.

6.3.2 ***Performance Against SGM.*** Fig. 8b, shows only one line representing SGM. This is because SGM only produces one Global Model, and unlike Local-non-IID, where each client gets a different distribution, Global-IID is a single test set for all the models. Hence, only one accuracy result. Even with all the 50 clients combined, the performance of SGM on IID data is poor. This is because of the existence of some clients with extremely skewed distribution. Such skewed models may negate the impact of the other clients, leading to poor performance. Even here, we can notice that our approach has an almost linear improvement in performance, owing to decreasing uncertainty and increasing similarity over rounds. This uncertainty decrease increases the appropriate contribution of many clients, leading to improved Global-IID performance.

6.3.3 ***Performance Against CM.*** Fig. 8c shows that similar to Local-non-IID performance, the CM seems to be performing better in the early rounds of training. But, as the number of rounds increases, our approach performs better than CM. In clustering-based models, only the clients in the cluster can contribute to their CM.

Table 1: Comparing the prediction accuracy of the client models trained with different SOTA non-IID FL methods. The results are based on the top-5 accuracy in round 200. DD1 and DD2 represents Dirichlet Distribution coefficient of 0.6, and 0.3 respectively.

Top-5 / Method	DD1			DD2		
	minAcc	maxAcc	avgAcc	minAcc	maxAcc	avgAcc
FedProx [29]	0.581	0.818	0.650	0.548	0.731	0.622
FedDC [7]	0.674	0.901	0.724	0.611	0.893	0.693
FedEM [20]	0.707	0.891	0.790	0.687	0.897	0.753
ClusteredFL [30]	0.658	0.854	0.700	0.559	0.713	0.662
SLFed (Ours)	0.741	0.935	0.863	0.735	0.943	0.879

This restriction causes some clustered models to not have any features related to the clients on other clusters. But, in our approach, as the uncertainty decreases, more rescaled models get added to the PAM, even if it is not highly similar. Over time, this leads to each of the models having a much better overall performance.

From Fig. 7b and Fig. 8b, we can notice that quick performance boost in our method against Local-non-IID, but the improvement for Global-IID data is much more linear. In the earlier training rounds, the models with high similarity contribute the most to the PAM, leading to better Local-non-IID performance. But, as the number of rounds increases and the overall uncertainty decreases, the not-so-similar models also make small contributions to the PAM, causing a gradual improvement in Global-IID performance.

6.4 Comparison with State-of-the-Art Methods

This subsection will compare our approach against state-of-the-art FL methods designed to work well for non-IID datasets. For this test, we use the ImageNet32 dataset distributed under two Dirichlet distributions: 0.6 and 0.3 [7]. We select methods like [7, 20, 29, 30] for this comparison. FedDC [7] introduces lightweight modifications in the local training phase to track the gap between local and global parameters using auxiliary drift variables. FedEM [20] treats the local data distribution of the clients to be unknown underlying distribution and builds personalized models for clients not seen in training time. ClusteredFL [30] exploits geometric properties of the loss surface in order to cluster the client population into different groups. FedProx [29] is a reparameterized version of the famous FedAvg approach and is designed to work well with heterogenous data distributions.

For our experiment, we will be considering the performance of the methods when tested with local non-IID datasets. The results will be presented as minAcc, maxAcc, and avgAcc. The minAcc and maxAcc denote the client model, which yields the minimum, and the client model, which yields the maximum accuracy, respectively. The avgAcc is the average performance accuracy of all the 100 models.

In Table. 1, we compare the results of the Top-5 accuracy of all the models in round 200. We can see that our approach's performance is better than existing works when we consider the model's performance against the client's local data distribution. This shows that each client successfully manages to find and aggregate the models that are similar to it. This is a 40-class classification problem with a 32x32 dataset. Yet, our method gets a high average accuracy across all models. When the Dirichlet co-efficient is 0.3, it leads to highly uneven distribution. Even in that case, our model outperforms existing SOTA approaches.

7 DISCUSSIONS

7.1 Computational Complexity

Let's discuss the computational overhead induced by using GAN-based similarity checking. The primary concern would be that generating images using GAN might be costlier than the model training. This case is untrue because we do not train the GAN models to exhibit efficient performance. The GAN only needs to be trained for a few iterations to get images that reflect the distribution of the client. Individually, the generated images do not require high prediction accuracy, drastically reducing the computational overhead.

Reducing Computation Complexity. The GAN images need not be generated in each round. Initially, when the original model performance is improving rapidly, the GAN images need to be generated once every 3-4 rounds. But, as the model performance stabilizes, the same GAN images can be used for multiple rounds rather than regenerating newer images.

7.2 Information Leakage

Another concern surrounding the use of GAN can be attributed to the possibility of information leakage because of the GAN images that are sent to other clients. As mentioned earlier, the GAN is not trained long enough for the features of the local dataset to be captured in the images. From the example shown in 5, we can notice that there are no details leaked that can be perceived by human beings because of how the images are in the early stages of the GAN training process.

Enhancing Privacy. Instead of using vanilla GAN, the clients can choose to use privacy-preserving GAN methods like [19, 32]. Such GAN methods specifically focus on reducing an attacker's ability to infer the data used for training the GAN model. Another approach to defending against information leakage would be to use methods like Locality Sensitive Hashing [14] to group the models based on their similarities. This approach would require a central

server to conduct the task and is not as granular as our decentralized approach.

7.3 Communication Efficiency

Another concern with our approach is communication efficiency. Our method has two processes not part of the general FL scheme, making it communicationally costlier than standard FL. The two processes are GAN-generated data sharing and layer-wise secret sharing. This communication overhead can be costly for some intricate FL tasks.

Improving Communication Efficiency. Suppose the clients find the communication cost to be high. In that case, we can adapt this decentralized scheme to function with a centralized server, such that the centralized server has no real power but just acts as a storage server. The clients can store their GAN data records on the server and grant access to other clients whenever needed. Similarly, the layer-wise secret sharing can be replaced by sending the local model, PAM_{list}, and σ directly to the central server. The central server can take care of the aggregation process. It is essential to ensure the trustworthiness of the central server if this technique is used.

7.4 Future Work

Our framework is safe and efficient as long as the clients don't deviate from the protocol but is susceptible to security attacks under other conditions. In the future, we intend to explore the possibility of using a Multi-Armed Bandit [1] for the DS selection process. We also want to make our approach data-type agnostic. Clients with local models from different data domains like images, audio, video, and text should all be part of the same safe and private FL network and still build the PAM suitable for them.

Centralized Variation. A centralized variation of our approach can utilize the global view to quicken the client evaluation process. It can also improve the network's security by applying clipping methods and adding noise to make PAMs differentially private. Each client sends all their requests and information through the trusted central server. Multiple trusted central servers can also be introduced into the network to reduce the individual computation and communication overhead.

8 CONCLUSION

In this work, we have explored the possibility of establishing individual Personal Aggregated Models for all the clients through a Subjective Logic-based decentralized learning approach. We use a GAN-generated dataset to discover training data similarities between clients and utilize direct and indirect client information to select contributors for the Personal Aggregated Model. We employ a Dynamic Uncertainty Threshold for shortlisting the clients in the network for aggregation. Our layer-wise secret sharing allows clients to share models among themselves without compromising privacy. We also show the superior performance of our approach against Single Global Model methods and Cluster Model methods under both IID and non-IID settings. Our approach outperforms the state-of-the-art non-IID FL methods when tested on each client's local data distribution. Finally, we also discuss the communication, computation, and privacy improvements to our method.

ACKNOWLEDGMENTS

This work is partially supported by the National Science Foundation under Grant Numbers DGE-2011117, DGE-2146359, CNS-2349233, and CNS-1852105; the National Natural Science Foundation of China under Grant 62002059; and the Fundamental Research Funds for the Central Universities under Grant 2242022k60005.

REFERENCES

[1] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. 2002. Finite-time analysis of the multiarmed bandit problem. *Machine learning* 47, 2 (2002), 235–256.
[2] Christopher Briggs, Zhong Fan, and Peter Andras. 2020. Federated learning with hierarchical clustering of local updates to improve training on non-IID data. In *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 1–9.
[3] Haokun Chen, Ahmed Frikha, Denis Krompass, Jindong Gu, and Volker Tresp. 2023. FRAug: Tackling federated learning with Non-IID features via representation augmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 4849–4859.
[4] Yujing Chen, Yue Ning, Martin Slawski, and Huzefa Rangwala. 2020. Asynchronous online federated learning for edge devices with non-iid data. In *2020 IEEE International Conference on Big Data (Big Data)*. IEEE, 15–24.
[5] Patryk Chrabaszcz, Ilya Loshchilov, and Frank Hutter. 2017. A downsampled variant of imagenet as an alternative to the cifar datasets. *arXiv preprint arXiv:1707.08819* (2017).
[6] Clement Fung, Chris JM Yoon, and Ivan Beschastnikh. 2020. The limitations of federated learning in sybil settings. In *23rd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2020)*. 301–316.
[7] Liang Gao, Huazhu Fu, Li Li, Yingwen Chen, Ming Xu, and Cheng-Zhong Xu. 2022. Feddc: Federated learning with non-iid data via local drift decoupling and correction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 10112–10121.
[8] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. *Advances in neural information processing systems* 27 (2014).
[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.
[10] István Hegedűs, Gábor Danner, and Márk Jelasity. 2019. Gossip learning as a decentralized alternative to federated learning. In *IFIP International Conference on Distributed Applications and Interoperable Systems*. Springer, 74–90.
[11] Chenghao Hu, Jingyan Jiang, and Zhi Wang. 2019. Decentralized federated learning: A segmented gossip approach. *arXiv preprint arXiv:1908.07782* (2019).
[12] Hongsheng Hu, Zoran Salcic, Lichao Sun, Gillian Dobbie, Philip S Yu, and Xuyun Zhang. 2022. Membership inference attacks on machine learning: A survey. *ACM Computing Surveys (CSUR)* 54, 11s (2022), 1–37.
[13] Md Sirajul Islam, Simin Javaherian, Fei Xu, Xu Yuan, Li Chen, and Nian-Feng Tzeng. 2024. FedClust: Optimizing Federated Learning on Non-IID Data through Weight-Driven Client Clustering. *arXiv preprint arXiv:2403.04144* (2024).
[14] Omid Jafari, Preeti Maurya, Parth Nagarkar, Khandker Mushfiqul Islam, and Chidambaram Crushev. 2021. A survey on locality sensitive hashing algorithms and their applications. *arXiv preprint arXiv:2102.08942* (2021).
[15] Audun Jøsang. 1997. Artificial reasoning with subjective logic. In *Proceedings of the second Australian workshop on commonsense reasoning*, Vol. 48. Citeseer, 34.
[16] Audun Jøsang. 2001. A logic for uncertain probabilities. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 9, 03 (2001), 279–311.
[17] Xiaoxiao Li, Meirui Jiang, Xiaofei Zhang, Michael Kamp, and Qi Dou. 2021. Fedbn: Federated learning on non-iid features via local batch normalization. *arXiv preprint arXiv:2102.07623* (2021).
[18] Yuzheng Li, Chuan Chen, Nan Liu, Huawei Huang, Zibin Zheng, and Qiang Yan. 2020. A blockchain-based decentralized federated learning framework with committee consensus. *IEEE Network* 35, 1 (2020), 234–241.
[19] Yi Liu, Jialiang Peng, JQ James, and Yi Wu. 2019. PPGAN: Privacy-preserving generative adversarial network. In *2019 IEEE 25Th international conference on parallel and distributed systems (ICPADS)*. IEEE, 985–989.
[20] Othmane Marfoq, Giovanni Neglia, Aurélien Bellet, Laetitia Kameni, and Richard Vidal. 2021. Federated multi-task learning under a mixture of distributions. *Advances in Neural Information Processing Systems* 34 (2021), 15434–15447.
[21] Leland McInnes, John Healy, and Steve Astels. 2017. hdbscan: Hierarchical density based clustering. *J. Open Source Softw.* 2, 11 (2017), 205.
[22] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*. PMLR, 1273–1282.
[23] Mahdi Morafah, Saeed Vahidian, Weijia Wang, and Bill Lin. 2023. Flis: Clustered federated learning via inference similarity for non-iid data distribution. *IEEE*

Open Journal of the Computer Society 4 (2023), 109–120.

[24] Xutong Mu, Yulong Shen, Ke Cheng, Xueli Geng, Jiaxuan Fu, Tao Zhang, and Zhiwei Zhang. 2023. Fedproc: Prototypical contrastive federated learning on non-iid data. *Future Generation Computer Systems* 143 (2023), 93–104.

[25] Thien Duc Nguyen, Phillip Rieger, Huili Chen, Hossein Yalame, Helen Möllering, Hossein Fereidooni, Samuel Marchal, Markus Miettinen, Azalia Mirhoseini, Shaza Zeitouni, et al. 2022. FLAME: Taming Backdoors in Federated Learning. (2022).

[26] Christodoulos Pappas, Dimitris Chatzopoulos, Spyros Lalis, and Manolis Vavalis. 2021. Ipls: A framework for decentralized federated learning. In *2021 IFIP Networking Conference (IFIP Networking)*. IEEE, 1–6.

[27] Phillip Rieger, Thien Duc Nguyen, Markus Miettinen, and Ahmad-Reza Sadeghi. 2022. DeepSight: Mitigating Backdoor Attacks in Federated Learning Through Deep Model Inspection. *arXiv preprint arXiv:2201.00763* (2022).

[28] Abhijit Guha Roy, Shayan Siddiqui, Sebastian Pölsterl, Nassir Navab, and Christian Wachinger. 2019. Braintorrent: A peer-to-peer environment for decentralized federated learning. *arXiv preprint arXiv:1905.06731* (2019).

[29] Anit Kumar Sahu, Tian Li, Maziar Sanjabi, Manzil Zaheer, Ameet Talwalkar, and Virginia Smith. 2018. On the convergence of federated optimization in heterogeneous networks. *arXiv preprint arXiv:1812.06127* 3 (2018), 3.

[30] Felix Sattler, Klaus-Robert Müller, and Wojciech Samek. 2021. Clustered Federated Learning: Model-Agnostic Distributed Multitask Optimization Under Privacy Constraints. *IEEE Transactions on Neural Networks and Learning Systems* 32, 8 (2021), 3710–3722. https://doi.org/10.1109/TNNLS.2020.3015958

[31] Adi Shamir. 1979. How to share a secret. *Commun. ACM* 22, 11 (1979), 612–613.

[32] Warit Sirichotedumrong and Hitoshi Kiya. 2021. A gan-based image transformation scheme for privacy-preserving deep neural networks. In *2020 28th European Signal Processing Conference (EUSIPCO)*. IEEE, 745–749.

[33] Yue Tan, Yixin Liu, Guodong Long, Jing Jiang, Qinghua Lu, and Chengqi Zhang. 2023. Federated learning on non-iid graphs via structural knowledge sharing. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 37. 9953–9961.

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

[35] Yanmeng Wang, Qingjiang Shi, and Tsung-Hui Chang. 2023. Why batch normalization damage federated learning on non-iid data? *arXiv preprint arXiv:2301.02982* (2023).

[36] Ming Xie, Guodong Long, Tao Shen, Tianyi Zhou, Xianzhi Wang, Jing Jiang, and Chengqi Zhang. 2020. Multi-center federated learning. *arXiv preprint arXiv:2005.01026* (2020).

[37] Lei Yang, Jiaming Huang, Wanyu Lin, and Jiannong Cao. 2023. Personalized federated learning on non-IID data via group-based meta-learning. *ACM Transactions on Knowledge Discovery from Data* 17, 4 (2023), 1–20.

[38] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. 2018. Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582* (2018).