FedReG: Recouping the Global Model in Personalized Federated Learning

Tianyi Liu
Delft University of Technology
Delft, The Netherlands
ltylty221@gmail.com

Mingkun Yang
Delft University of Technology
Delft, The Netherlands
m.yang-3@tudelft.nl

Qing Wang
Delft University of Technology
Delft, The Netherlands
qing.wang@tudelft.nl

ABSTRACT

With the widespread application of artificial intelligence, centralized machine learning approaches, which require access to users' local data, have raised concerns about data privacy. In response, federated learning (FL) has been proposed where models are trained locally on user data and then offloaded to the central server for global aggregation. However, a single global model in FL struggles to meet the diverse personalized needs of clients and suffers significant accuracy degradation when client data distributions are uneven or exhibit non-IID characteristics. Personalized federated learning has been introduced to address data heterogeneity and to meet customized needs. Yet, personalized federated learning (PFL) also has shortcomings: the global model in federated learning often becomes an intermediary product in this framework, lacking the advantage of learning a generalized model. In this paper, we propose **FedReG**, a new scheme in personalized federated learning with parameter decoupling and rebalanced dataset, to recoup the performance of the global model. By leveraging parameter decoupling and introducing a rebalanced dataset generated according to the distribution of clients' local data, FedReG achieves high accuracy for both the global model and average client models. Extensive experiments also demonstrate the superior scalability of FedReG and robustness over other federated learning and personalized federated learning algorithms.

CCS CONCEPTS

• **Computing methodologies** → *Machine learning*; *Distributed algorithms*; **Distributed artificial intelligence**.

KEYWORDS

Personalized federated learning, parameter decoupling, data augmentation

1 INTRODUCTION

Artificial Intelligence (AI) technologies is now mature and has also been integrated into various applications, reshaping numerous industries. This growth has been significantly bolstered by the development of intelligent devices, such as smartphones, smartwatches, and smart gateways. These devices, with their enhanced capabilities, have contributed to an exponential increase in data, which is crucial for AI-driven solutions [32]. However, the rising concerns over data privacy have led legislative bodies in various countries to implement measures such as the European General Data Protection Regulation (GDPR) [27] and the California Consumer Privacy Act (CCPA) [12], presenting new challenges to machine learning solutions that leverage big data and increasing demands for privacy-preserving AI [8].

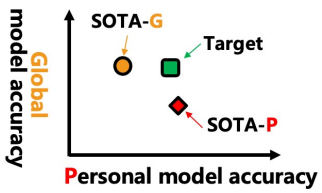

Figure 1: The motivation of our FedReG. We target recouping the performance of the global model in personalized federated learning.

Federated Learning (FL) emerged as a solution to tackle these privacy concerns [15, 26]. Introduced by McMahan et al. in 2016 [20], FL began with the Federated Averaging (FedAvg) algorithm. Unlike traditional approaches, FL involves training local models on individual devices and then aggregating these models on a central server to form a global model. This process ensures that personal data remains on user's device, thereby enhancing data security [13, 19].

Despite these advantages, Federated Learning faces challenges in practical applications, especially when dealing with non-independent and identically distributed (non-IID) data. In such scenarios, where client data distributions are unbalanced and heterogeneous, traditional FL approaches often perform suboptimally [18, 26, 31].

To address the performance degradation of traditional federated learning algorithms due to data heterogeneity and to meet clients' personalized needs, a research direction known as *Personalized Federated Learning* (PFL), has gradually gained attention. Personalized Federated Learning aims to provide participating clients in federated learning with a personalized model that performs better, rather than providing only a single global model. However, many personalized federated learning algorithms, such as FedPer [6], FedRep [2], and FedBABU [23], often result in *decreased performance of the global model*, to achieve better personalized models.

In this paper, we target recouping the performance of the global model in personalized FL, as illustrated in Figure 1. We propose a new training architecture **FedReG**, based on model decoupling in personalized FL as well as by generating a rebalanced dataset according to the distribution of clients' local data. FedReG seeks to recoup the gain in the global model's performance within personalized FL algorithms while maintaining as well the performance of clients' personalized models. To evaluate FedReG, we compare it with nine other state-of-the-art (SOTA) methods on four different datasets and analyze the algorithm's scalability, and robustness. Our main contributions are summarized below:

- We propose a training framework, FedReG, for personalized FL based on model decoupling. This approach creates a new rebalanced dataset from the user's original local data without

requiring additional data sample exchanges with the server. This allows the server to generate a well-performing global model while maintaining local accuracy for users.
- We explore the impact of the size and creation method of the rebalanced dataset required by FedReG during training and find an optimal solution that achieves the highest global model accuracy across various configurations. FedReG also modifies the aggregation weights during the aggregation process at the server to further increase the accuracy of the global model.
- We build a testbed with 10 Jetson Nanos to validate the performance of FedReG and compare it with nine state-of-the-art methods. The results have demonstrated the effectiveness of our proposed FedReG in practice, in terms of accuracy, scalability, and robustness.

2 BACKGROUND

2.1 Federated Learning

Federated Learning (FL), first conceptualized by the authors of FedAvg [20], aims to train a global model that performs well across multiple clients, or an "average client" [26]. This contrasts with traditional approaches that rely on transferring local user data to a centralized server for training[15]. Figure 2 depicts the training process of the traditional FL framework, also described below:

a) At the beginning of a communication round, the server sends a global model to all the clients that have been selected for participation in that round.
b) The selected clients train the received global model on their local data.
c) Upon completing training, these clients send the parameters of their trained models back to the server.
d) The server aggregates these parameters and updates them into a new global model for use in the next round.

The goal of each round is to obtain an optimal global model w that minimizes the aggregated local loss function $f_m(w_m)$ [32]:

$$f_m(w_m) = \frac{1}{D_m} \sum_{i}^{D_m} l(x_i, y_i; w_m) \ ; \tag{1}$$

$$\min f(w) = \sum_{m=1}^{M=\lfloor C \times K \rfloor} \frac{D_m}{D} f_m(w_m) \ . \tag{2}$$

Here, x_i and y_i represent the features and labels of sample i , respectively. D_m is the size of the client's local dataset, D is the total size of samples from all clients participating in the round, C is the participation rate per round, K is the total number of clients, m is the ID of a client that joins in this round, and l is the loss function.

2.2 Personalized Federated Learning

Personalized FL is a research direction built upon the foundation of FL. It aims to overcome the challenges posed by data heterogeneity while catering to the personalized needs of users [25]. Personalized FL achieves its goal by personalizing the global model or by allowing users to learn their personalized models locally [26]. This ensures that each model performs well locally for all participating users. There are various approaches to personalized FL, such as data augmentation, client selection, regularizing local loss, or data decoupling. Below, we briefly introduce data decoupling solutions in personalized FL that are mostly related to our FedReG.

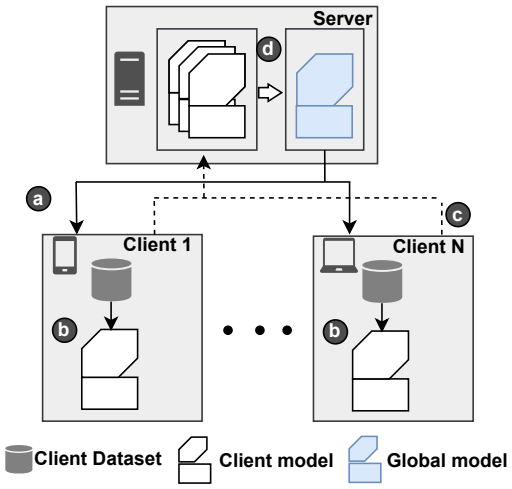

Figure 2: Process of the traditional federated learning algorithms in one communication round.

Parameter Decoupling. FedPer [6] introduces a novel training framework where the model is partitioned into *base layers* and *personalization layers*, also referred to as the model's *head*. This model structure draws inspiration from multi-task learning and transfer learning concepts. The base layers, akin to the shared layers in multi-task learning (MTL), capture features from various data types [24, 30]. Meanwhile, the personalization layers, similar to task-specific layers in multi-task learning, learn features and data distribution unique to each user [26]. Similar implementations, dividing the model into base and head, have also been utilized in methods such as FedRep [2], FedBABU [23], GPFL [29], and FedRoD [7], etc. These approaches reflect a common understanding of the need for both shared knowledge and individual personalization within the paradigm of FL.

3 FEDREG DESIGN

3.1 Overview

Figure 3 illustrates the overall training process of our FedReG. To recoup the performance of global model in personalized FL, we design at each client an additional model head, *HeadG*, which is sent to the FL server for aggregation. Also, we derive a *Rebalanced Dataset* on the client's local data to train the model base and *HeadG* for global model aggregation.

The client's model base and *HeadP* form a complete client model, and *HeadP* is kept on each client. *HeadG* has the same structure as *HeadP*, which are usually the last few linear layers of the model in the approach of parameter decoupling.

Rebalanced dataset. The rebalanced dataset is introduced based on the fact that the local models trained on the FL clients will benefit the global model at the server if they are trained on iid (i.e., balanced) data.

To build such a rebalanced dataset, each class within the local dataset of a client is equalized either by augmenting the data from the original dataset through data augmentation techniques or by

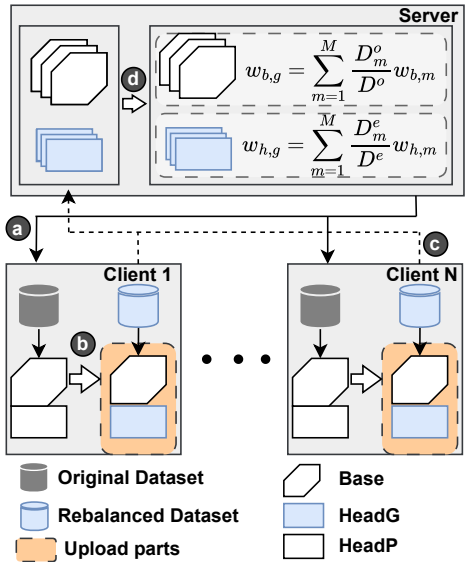

Figure 3: The procedure of the proposed FedReG in one communication round.

randomly selecting to reduce to the same value. Each client's rebalanced dataset retains the same classes as the original dataset, without acquiring additional data not belonging to the original dataset through communication with other clients or the FL server. Figure 4 gives an example, illustrating the data distribution of the original datasets of 30 clients and their rebalanced datasets.

Following the traditional FL process, the process of our FedReG are shown in Figure 3 and described as follows:

- At the beginning of each communication round, the server sends the global model to all clients that have been selected for participation in this round.
- The selected clients train the local model base and *HeadP* retained locally with the original dataset and then train the local model's base and *HeadG* used for global model aggregation with the rebalanced dataset.
- Clients upload the parameters of the local model's base and *HeadG* and related data used, including the client's original dataset size D_m^o and the effective sample size D_m^e, for aggregation to the server.
- The server calculates the weights for aggregating the base and *HeadG* separately based on client's original dataset size D_m^o and the effective sample size D_m^e in the rebalanced dataset; then it aggregates the global model.

The derivation of the effective sample size D^e and D_m^e shown in Figure 3 will be explained in Section 3.4.

3.2 Local Training on FedReG Client

As shown in Figure 5, the local training process of the user consists of two main steps, with the model trained sequentially through the original dataset and the rebalanced dataset.

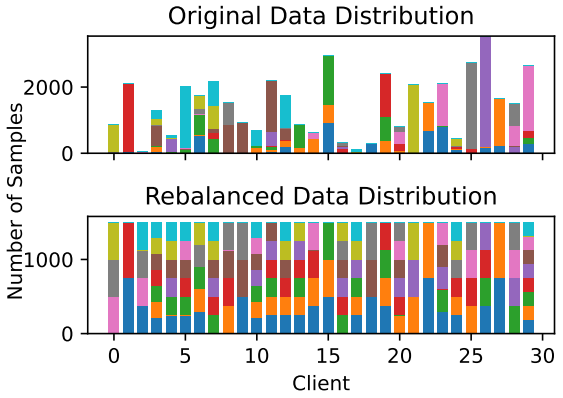

Figure 4: The illustration of the data distribution of the original dataset and rebalanced dataset, with each color denoting the data of a distinct class.

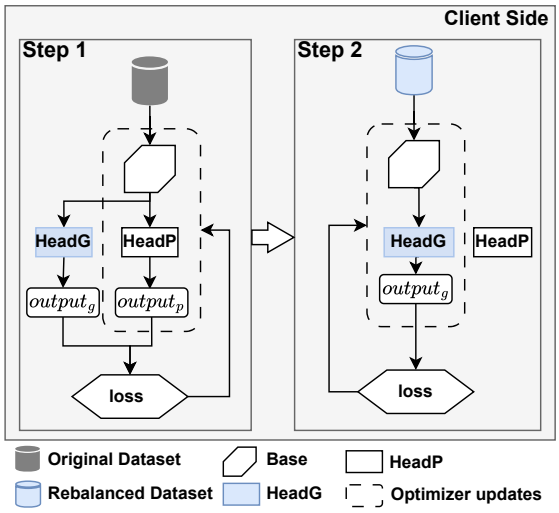

Figure 5: The local training at the FedReG client.

- Training starts with the original dataset, where data is processed through the base layer to output an intermediate result, *mid*. The *mid* is then processed through *HeadG* and *HeadP*, respectively outputting $output_g$ as a prediction result with a more balanced distribution among classes and $output_p$ as a prediction result where the distribution among classes depends on the local data distribution. "$output_p + output_g$" is used as the output result for input into the loss function to calculate the gradient. During backpropagation, the gradient is propagated back to the base through both *HeadG* and *HeadP*. Subsequently, the optimizer updates the parameters of the Base and Head *HeadP*, while *HeadG* remains unchanged. The idea of using "$output_p + output_g$" as the output result is to allow Head *HeadP*, which is closer to the local data distribution, to be trained more quickly with the help of *HeadG*, which is aggregated through FL and has a more balanced and generalizable prediction for various classes.

Algorithm 1: FedReG Client

Input: loss function f, Optimizer, Client Local Dataset $d_{o,m}$, Rebalance Dataset $d_{r,m}$

Initialize client m local weight w_m at random
for *communication round k = 1,2,3,...* **do**
  if *Client m is selected* **then**
    Receive $w_g^{(k)}$ from Server
    for *batch* $b_o = (x_o, y_o)$ *in* d_o **do**
      mid ← base(x_o)
      $output_g$ ← $HeadG$(mid)
      $output_p$ ← $HeadP$(mid)
      loss ← f($output_g$+$output_p$,y_o)
      loss function backwards
      Optimizer updates $w_{b,m}^{(k)}$,$w_{hp,m}^{(k)}$
    end
    for *batch* $b_r = (x_r, y_r)$ *in* d_r **do**
      mid ← base(x_r)
      $output_g$ ← $HeadG$(mid)
      loss ← f($output_g$,y_r)
      loss function backwards
      Optimizer updates $w_{b,m}^{(k)}$,$w_{hg,m}^{(k)}$
    end
    Send D_m^o,D_m^r
    Send $w_{b,m}^{(k)}$, $w_{hg,m}^{(k)}$ to Server
  end
end

- Next, the rebalanced dataset (cf. Section 3.4 for the construction of this dataset) is used to retrain the base. The output *mid* result is only fed to $HeadG$, and the loss is calculated only based on $HeadG$'s $output_g$. During backpropagation, it only propagates through $HeadG$ to the base. The optimizer then updates the base and $HeadG$. After this, the base and $HeadG$, as a complete model, upload their parameters to the server for aggregation. This step mainly aims to train $HeadG$ with a more balanced prediction result distribution through a rebalanced dataset with a more balanced data distribution, providing the server with a model that has strong generalizability. This step updates the base part because it is considered a low-level feature extractor, and the low-level features learned from the rebalanced dataset generated based on the original dataset are not significantly different from those learned from the original dataset. Moreover, the rebalanced dataset includes samples generated through data augmentation, which should make the base part more robust.

The pseudo-code of the client-side operations is presented in Algorithm 1. It is worth noting that the number of training samples for the local model's base part and $HeadG$ are actually different because the rebalanced dataset could be created with different threshold t. Therefore, some adjustments to the server aggregation step are proposed in FedReG, which will be detailed in the next subsection.

3.3 Federated Learning on FedReG Server

As shown in Figure 3, an explanation of all components in the entire process is provided. Each client holds their own original dataset, and a rebalanced dataset is generated from it. The model structure held by each client is the same, with each having a base layer of the model, and Head layers $HeadG$ and $HeadP$. By combining the base layer with the both head layers, the model can make complete predictions and learn about each category of a task or dataset, meaning the Head layers will output for every class of the dataset, regardless of whether the local user has data in those classes.

The Head layer $HeadG$ is trained with the local rebalanced dataset, resulting in $HeadG$ being influenced by a dataset that is closer to an IID distribution, leading to more balanced model predictions. When a certain class significantly exceeds others in quantity, machine learning models tend to predict future outcomes as the class with more data in training set.

However, by training $HeadG$ with the rebalanced dataset, the predictions made by combining the base layer and $HeadG$ will not be biased towards a specific class due to unbalanced local data distribution. Subsequently, $HeadG$, along with the base layer, is transmitted to the centralized server for aggregation, resulting in a model with strong generalizability.

The Head layer $HeadP$ is trained with the local non-IID distributed original data, learning the characteristics of the local data distribution. Thus, predictions made by combining the base layer and $HeadP$ will be more aligned with the local data distribution. After training, $HeadP$ is retained locally and not updated through federated learning. When predicting locally, both $HeadG$ and $HeadP$ work together.

During server aggregation, the aggregation weights differ from traditional FL. They are based on the size of the original datasets of the clients participating in the current round. Typically, the server's aggregation part adopts the FedAvg aggregation approach, written as follows:

$$w_g = \sum_{m=1}^{M} \frac{D_m}{D} w_m \,, \tag{3}$$

where w_g is the global model's parameters, w_m is the parameters of client m's model, D_m is the size of client m's original dataset, D is the total size of the original datasets of all clients participating in the current round, and M is the number of participating clients.

However, in FedReG, the model's base part and head part are aggregated separately with different weights, as follows:

$$w_{b,g} = \sum_{m=1}^{M} \frac{D_m^o}{D^o} w_{b,m} \,, \tag{4}$$

$$w_{h,g} = \sum_{m=1}^{M} \frac{D_m^e}{D^e} w_{hg,m} \,, \tag{5}$$

where $w_{b,m}$ is the weight of each user's base, $w_{b,g}$ is the weight of the aggregated global model's base, $w_{hg,m}$ is the weight of each client's $HeadG$, and $w_{h,g}$ is the weight of the aggregated global model's head. D_m^o is the size of client m's original dataset, which is the previously mentioned D_m. D^o is the same as D, the total size of the original datasets of all clients participating in the current round. D_m^e refers to the number of effective samples in client m's

Algorithm 2: FedReG Server

Initialize global model weight w_g at random **for** *Communication round k = 1,2,3,...* **do**
  Select a set of Clients M joining in this round
  Send $w_g^{(k)}$ to Client $m \in M$
  Receive D_m^o,D_m^r
  Receive $w_{b,m}^{(k)}$, $w_{hg,m}^{(k)}$ from Client $m \in M$
  Calculate Clients' aggregation Weight $\gamma_{b,m} = \frac{D_m^o}{D^o}$, $\gamma_{h,m} = \frac{D_m^r}{D^r}$
  Aggregate global model base with $w_{b,g}^{(k+1)} = \sum_{m=1}^{M} \gamma_{b,m} \times w_{b,m}^{(k)}$
  Aggregate global model head with $w_{h,g}^{(k+1)} = \sum_{m=1}^{M} \gamma_{h,m} \times w_{hg,m}^{(k)}$
end

rebalanced dataset. D^e is the total number of effective samples in the rebalanced datasets of all clients participating in current round.

The effective samples are the number of samples obtained in the rebalanced dataset that are not generated through data augmentation. If a client has a target value t_c, which will be explained in next subsection, then the number of effective samples is calculated as follows:

$$D_m^e = \sum_{i}^{k} e_i \,, \tag{6}$$

where k is the total number of classes, i is the class label, and e_i is the number of effective samples in class i in the rebalanced dataset. For classes in the original dataset with fewer samples than t_c, e_i is the number of samples in class i, as the other data are generated through data augmentation. For classes with equal to or more than t_c, e_i is t_c as the data for these classes are reduced to t_c.

Algorithm 2 presents the pseudo-code for server-side operations of FedReG.

3.4 Rebalanced Dataset

The rebalanced dataset is generated from the client's local original dataset by first setting a threshold t to determine the size of each client's rebalanced dataset. This threshold could be the mean, median, or any other statistical measure derived from the size of all clients' original datasets. Once fixed, each client calculates the number of data samples needed per class t_c based on the number of classes i they have, given by

$$t_c = t/i \,. \tag{7}$$

Upon establishing this value, classes with samples exceeding t_c will randomly select t_c samples, while those with fewer than t_c samples will employ data augmentation techniques to increase their count to t_c.

The primary goal is to locally train a more balanced model head *HeadG*, which in turn contributes to a more balanced and generic model through federated learning aggregation. Figure 4 (cf. Section 3.1) depicts the distribution of the rebalanced dataset among clients when the threshold t is chosen to be the average number of client samples. The original dataset is produced based on the Dirichlet distribution.

Algorithm 3: Generating Rebalanced Dataset

Input: Client Local Dataset $d_{o,m}$, threshold t
Initial empty Rebalanced Dataset $d_{r,m}$
Calculate t_c base on local dataset
for *Class label* $i = 0, 1, 2, \cdots, k$ **do**
  $n_i \leftarrow$ Count quantity of data of class i
end
for *Class label* $i = 0, 1, 2, \cdots, k$ **do**
  if $0 < n_i < t_c$ **then**
    Generate $t_c - n_i$ augmented data base on existed class i data
    Add augmented data to the Rebalanced Dataset $d_{r,m}$
    Add original data fo class i to the Rebalanced Datset $d_{r,m}$
    $e_i \leftarrow n_i$
  else
    if $n_i \geq t_c$ **then**
      Random Select t_c data from class i
      Add selected data to the Rebalanced Dataset $d_{r,m}$
      $e_i \leftarrow t_c$
    end
  end
end

Algorithm 3 presents the pseudo-code for the generation of the rebalanced dataset.

4 PERFORMANCE EVALUATION

This section will first introduce the baseline algorithms used for comparison. A description of the used datasets and the implemented experimental configurations will follow. Subsequently, the section will explore the optimal configuration of generating the rebalanced dataset for achieving the best performance with FedReG, including an analysis of the impact of two key factors: the augmentation method used to generate the rebalanced dataset and threshold t. Finally, FedReG will be compared with other baseline algorithms under the optimal configuration, and the advantages of FedReG over these algorithms are analyzed.

4.1 Setup

The model adopts the ConvNet used in FedDyn [5] and FedRoD [7], consisting of two convolutional layers, one pooling layer after each convolutional layer, and three fully connected layers, with the activation function of the fully connected layers being the ReLU function.

The used datasets are Cifar10, Cifar100 [14], EMNIST [9], and FMNIST [28]. The datasets' training and test sets are first mixed and then distributed using a Dirichlet distribution with α values of 0.1 and 0.5. Of the distributed data, 75% is used as the training set for the clients, and 25% as their test set. All clients' test sets are combined to form the global model's test set.

The communication rounds are set to 100, with 5 local training rounds for clients, a batch size of 20, an optimizer learning rate of 0.01, and a momentum of 0.9. Our reported performance results are the averages of five runs with the same experimental configuration. In each run, the highest global model accuracy (best global model accuracy) and every client accuracy (best average client accuracy) out of 100 rounds are recorded and then calculated their average values.

The global model and all client local models are tested after the FL server sends out the global model. The average client accuracy is actually calculated by summing the number of correctly predicted samples and the total number of samples in the test set, then dividing the two. The global model is tested using the combined test sets of all clients.

For the global model of GPFL[29], we have taken an additional step of aggregating the generic conditional input and personalized conditional input from clients as per the provided code[4]. This aggregation is performed on the server to construct a complete global model.

In comparison algorithms, PFL algorithms like FedPer, FedRep, and FedBABU do not have a complete global model for prediction. The entire local models of clients are aggregated during the aggregation process, and after distributing the global model, clients use parts of the parameters according to their algorithms. In traditional FL approaches, local model parameters are identical to global model parameters. Therefore, for these traditional FL approaches, the accuracy of the model distributed to the local and tested on the local test set (average client accuracy) is mostly the same as the global model accuracy in most cases.

4.2 Baselines

The evaluation mainly compares two types of FL approaches. The first type includes traditional FL schemes such as FedAvg [20], FedProx [17], and FedDyn [5]. The second type involves PFL schemes like Ditto [16], FedPer [6], FedRep [10], FedBABU [23], GPFL [29], and FedRoD [7]. Among these, FedPer, FedRep, FedBABU, GPFL, and FedRoD are algorithms that, similar to this paper, implement PFL through parameter decoupling. We also include the hyperparameter configurations used for different algorithms.

- **FedAvg (Federated Averaging)** [20]: As mentioned earlier, it is the earliest FL algorithm. It works by clients learning local models and then aggregating these models on a centralized FL server.
- **FedProx (Federated Proximity)** [17]: The algorithm introduces an additional proximal term in the local optimization problem to aid in the stability and improvement of the algorithm's convergence, especially in the face of data heterogeneity caused by non-IID data distribution. In the experiments, μ is set to 0.001, an option used in the original paper [1], and also used in FedROD [7] for replication and comparison.
- **FedDyn (Federated Dynamics)** [5]: The algorithm introduces a dynamic regularizer in the local loss function to align local model updates more closely with the global model's optimal solution.The strength of the dynamic regularizer, α, in the algorithm is set to 0.01 in the experiments, referring to the source code [3].

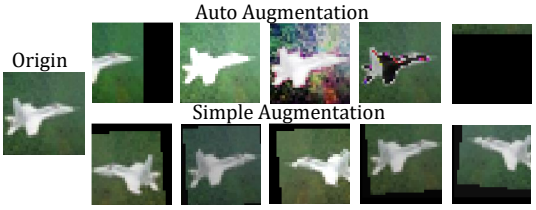

Figure 6: Data augmentation on Cifar10 data sample with auto augmentation and simple augmentation.

- **Ditto** [16]: A PFL scheme implemented through regularization. It aims to balance a single global model in FL and a personalized model trained locally without any FL scheme. The training rounds of the local personalized model become an extra hyperparameter, termed personal local step. It is set to 5 in the experiments.
- **FedPer (Federated Personalization)** [6]: A PFL algorithm and one of the main reference algorithms for the paper, implemented through parameter decoupling. The model is split into a base shared with the FL server and a head retained locally for personalization, allowing the model to accommodate different clients' personalized needs more effectively.
- **FedRep (Federated Representation Learning)** [10]: A PFL algorithm that also decouples parameters, splitting the model into a base shared with the FL server and a locally retained head. The training rounds of the local personalized model's head are an extra hyperparameter, also represented by the variable personal local step, set to 5 in the experiments.
- **FedBABU (Federated Body & Batchnorm Update)** [23]: As a PFL scheme implemented through data decoupling, FedBABU similarly splits the model into a base and a head. Unlike FedPer, during training, only the base part is trained and aggregated, while the head part is fine-tuned with local data for local testing. This approach primarily enhances the learning capability of feature representation for image classification tasks in frameworks. The number of fine-tuning rounds of the head part was set to 10 in the experiments.
- **GPFL** [29]: GPFL represents a novel algorithm for PFL achieved through parameter decoupling. It introduces the Global Category Embedding layer (GCE) and Conditional Valve (CoV) to simultaneously learn global and personalized feature information on each client. This approach enables the algorithm to effectively capture both shared and individual characteristics of data across different clients. Hyperparameters for GPFL, as per the provided source code [4], are set to $\lambda = 0.001$ and $\mu = 0.1$.
- **FedRoD (Federated Robust Decoupling)** [7]: Another PFL scheme based on data decoupling and another main reference algorithm for the paper. It splits the model into a base and two identical heads, one for aggregation and one for local personalization, and trains the two heads with different loss functions.

4.3 Impact of Augmentation Method and Threshold t

In Table 1, "Auto" refers to the Auto Augmentation method with preset enhancement strategies for datasets like Cifar10, IMAGNET, and SVHN [11]. "Simple" denotes a basic data augmentation process

Table 1: Shows the influence of different Data Augmentation methods on the algorithm with different threshold t.

Method	t	Global Acc	Average Acc
Auto	Max	53.27%	87.81%
	Median	52.58%	89.77%
	Average	55.26%	88.90%
	Sec-Min	53.57%	90.08%
Simple	Max	59.37%	90.29%
	Median	55.84%	89.64%
	Average	60.40%	90.25%
	Sec-Min	54.66%	90.58%

comprising five different methods: random horizontal flipping, random cropping, random rotation, color jitter, and random affine. For datasets such as FMNIST and EMNIST, which consist of grayscale images, the color jitter step is omitted.

Table 1 illustrates the impact of different settings on the Cifar10 dataset with 20 participating clients and a participation rate of 0.2 per round. It also compares the effects of two data augmentation methods under various threshold t settings on the performance of the algorithm by analyzing the generated Rebalanced dataset.

The results in Table 1 suggest that rebalanced datasets generated using simpler data augmentation methods generally yield higher global model accuracy. This indicates that employing complex data augmentation methods to generate the rebalanced dataset does not necessarily guarantee better performance. This could be due to the Auto Augmentation-enhanced data introducing excessive noise for ConvNet, as demonstrated in Figure 6. The data generated through Auto Augmentation might cause significant feature loss in the original data, whereas Simple augmentation only slightly alters the images, retaining the object features within them.

Regarding the selection of threshold t, it tends to be an empirical choice. Nevertheless, it is evident from Table 1 that choosing the average number of client samples as t for generating the Rebalanced dataset enables FedReG to achieve higher global model accuracy. The choice of different data augmentation methods and thresholds t has a limited impact on improving average client accuracy. Although using the maximum client sample size as t under Simple augmentation could yield comparable performance, considering the training overhead, it is advisable to choose the average value as threshold t. This effectively doubles the volume of user data.

4.4 Comparison with the State of the Art

Table 2 shows that FedReG consistently outperforms others in global model accuracy under all tested conditions.

- Compared to other PFL algorithms that do not specifically optimize global model accuracy, such as Ditto, FedPer, FedRep, FedBABU, and GPFL, which focus on achieving higher average client accuracy, FedReG shows a significant improvement in global model accuracy. These algorithms generally have lower global model accuracy than traditional FL algorithms. The superiority of FedReG is most pronounced on the FMNIST dataset, with improvements of 4%, 13.8%, 26.94%, 7.86%, 34.38% over these algorithms.
- Compared to FedRoD, which is specifically designed to enhance global model performance, FedReG also demonstrates better global model accuracy in Cifar100, FMNIST, Cifar10, and EMNIST. While FedRoD shows superior global model accuracy over traditional FL algorithms in Cifar10 and EMNIST, it underperforms FedAvg on the more class-diverse Cifar100 dataset. FedReG, however, leads FedRoD in global model accuracy on these four datasets by 2.75%, 2.86%, 0.47%, 0.5%, respectively.
- Compared to traditional FL algorithms, FedReG not only retains the advantage in average client accuracy typical of PFL algorithms but also surpasses them in global model accuracy. Across different datasets, FedReG leads the highest-performing traditional FL algorithms in global model accuracy by 1.33%, 3.81%, 1.45%, 1.3%.

In terms of average client accuracy, FedReG also maintains a leading position, showing slight improvements relative to other personalized solutions. However, in Cifar100, FedReG's average client accuracy is slightly behind the newer algorithm GPFL, yet still ahead of all other remaining solutions.

4.5 Scalability Analysis

Table 3 displays the global model accuracy and average client accuracy of different algorithms when the Cifar10 dataset is partitioned among 30, 50, 100, 200, and 500 clients using the Dirichlet distribution. As the number of clients increases from 30 to 500, and the number of participants in each round ranges from 6 to 100, it's notable that while the join rate remains the same, the actual amount of data involved in training does not change. However, the data allocated to each client becomes less.

When the number of clients is relatively small (30 and 50), algorithms like FedRoD and Ditto exhibit higher global model accuracy compared to traditional FL algorithms. As the number of clients increases from 30 to 500, a significant decrease in global model accuracy is observed for algorithms. For instance, the global model accuracy of FedRoD and Ditto drops from 60.14% and 56.32% to 40.94% and 43.68%, respectively, losing 19.2% and 12.64%. This is lower than the global model accuracy of traditional FL algorithms. Traditional FL algorithms are less affected in such scenarios, with a smaller decline. For example, FedAvg's global model accuracy decreases by only 1.45%.

This could be attributed to the fact that in algorithms, apart from FedRoD, the global model often serves as an intermediate product. However, FedReG continues to achieve higher global model accuracy than all other traditional FL and algorithms, even as the number of clients increases, while maintaining the highest average client accuracy. Remarkably, even with 500 clients, the global model accuracy of FedReG stands at 54.42%, surpassing the highest among other algorithms, FedAvg, at 53.46%. Its average client accuracy is 87.77%, higher than the highest among other algorithms, at 86.23%.

In Figure 7, by visualizing the $\log(10 \times G) + \log(10 \times P)$ values under different numbers of clients, we can also observe the significant advantage of FedReG in terms of scalability compared to other state-of-the-art algorithms. It is evident that, apart from the traditional FL algorithm, the overall performance of the global model and personal model in FedReG consistently outperforms the PFL baselines.

Table 2: Comparisons of accuracy between various algorithms and FedReG. In the table, columns labeled 'G' and 'P' denote the highest global model accuracy and the best average accuracy of the personal model, respectively. The abbreviation 'Jr.' indicates the proportion of clients participating in training per communication round, while 'Alg.' represents the name of the algorithm. The data were tested across four different datasets: Cifar100, FMNIST, Cifar10, and EMNIST. The testing involved 50 clients, with data distribution managed via the Dirichlet distribution, setting α to 0.1. FedReG outperforms other algorithms in global model accuracy.

Dataset	Cifar100		FMNIST		Cifar10		EMNIST	
Jr.	0.2		0.2		0.2		0.2	
Alg.	G	P	G	P	G	P	G	P
FedAvg	22.74%	22.74%	84.27%	84.27%	56.67%	56.67%	83.69%	83.69%
FedProx	23.40%	23.40%	84.54%	84.54%	55.70%	55.70%	83.59%	83.59%
FedDyn	17.05%	17.05%	83.43%	83.43%	53.94%	53.94%	82.54%	82.54%
Ditto	22.71%	43.58%	84.35%	96.53%	55.89%	84.27%	83.38%	92.21%
FedPer	13.81%	48.67%	74.55%	97.32%	40.41%	86.27%	69.66%	95.02%
FedRep	6.21%	37.20%	61.41%	97.20%	41.96%	86.39%	52.34%	93.36%
FedBABU	20.13%	43.93%	80.49%	80.49%	45.23%	86.03%	80.47%	92.03%
GPFL	11.13%	**56.91%**	53.97%	94.38%	32.14%	83.66%	48.62%	94.88%
FedRoD	21.98%	48.13%	85.49%	97.27%	57.65%	87.32%	84.49%	95.59%
FedReG	**24.73%**	50.26%	**88.35%**	**97.52%**	**58.12%**	**87.59%**	**84.99%**	**95.84%**

Table 3: Scalability analysis. The table presents the scalability of different algorithms tested on the Cifar10 dataset, with the Dirichlet distribution parameter α set at 0.1 and a joining rate of 0.2. During the tests, the batch size is adjusted based on the number of clients to manage computational resources effectively and ensure fair comparison. Specifically, the batch size is set to 10 when testing with 100 clients, and it is reduced to 1 for tests involving 200 and 500 clients.

Client	30		50		100		200		500	
Alg.	G	P	G	P	G	P	G	P	G	P
FedAvg	54.91%	54.91%	56.67%	56.67%	49.50%	49.50%	53.97%	52.87%	53.46%	53.46%
FedProx	55.87%	55.87%	55.70%	55.70%	51.32%	51.32%	51.84%	51.84%	43.89%	43.89%
FedDyn	55.08%	55.08%	53.94%	53.94%	51.75%	51.75%	50.49%	50.49%	47.23%	47.23%
Ditto	56.32%	85.85%	55.89%	84.27%	49.39%	84.68%	52.23%	84.01%	43.68%	80.83%
FedPer	43.09%	88.01%	40.41%	86.27%	44.22%	87.93%	43.45%	87.18%	41.26%	84.49%
FedRep	42.15%	87.75%	41.96%	86.39%	36.25%	85.91%	37.10%	86.72%	31.94%	81.16%
FedBABU	50.36%	86.79%	45.23%	86.03%	39.42%	85.18%	46.17%	86.88%	38.79%	82.26%
GPFL	32.29%	84.19%	32.14%	83.66%	37.10%	84.31%	32.71%	84.81%	36.66%	85.75%
FedRoD	60.14%	87.99%	57.65%	87.32%	49.85%	88.38%	51.61%	88.84%	40.94%	86.23%
FedReG	**60.50%**	**88.45%**	**58.12%**	**87.59%**	**57.70%**	**89.20%**	**55.91%**	**88.90%**	**54.42%**	**87.77%**

4.6 Robustness Analysis

In robustness analysis, we evaluate the robustness of an algorithm's global model accuracy to changes in data distribution heterogeneity. This is quantified by measuring the decrease in global model accuracy as the Dirichlet α value decreases, under otherwise unchanged configurations.

Table 4 presents the performance of various algorithms on Cifar10 and FMNIST datasets with 100 clients, under different α values of the Dirichlet distribution. As observed in Table 4, with a Dirichlet α of 0.5 (indicating a relatively uniform data distribution among clients), the global model accuracy of FedReG on Cifar10 and FMNIST is 62.15% and 90.38%, respectively. When the Dirichlet α decreases to 0.1, the accuracies of FedReG on Cifar10 and FMNIST drop to 57.70% and 88.21%, respectively, showing decreases of 4.45% and 2.17%.

Comparatively, focusing primarily on the Cifar10 dataset, traditional FL algorithms such as FedAvg, FedProx, and FedDyn exhibit decreases of 10.7%, 9.12%, and 5.81%, respectively. Other algorithms like Ditto, FedPer, FedRep, FedBABU, GPFL, and FedRoD show declines of 11.43%, 9.71%, 7.47%, 10.23%, 28.21%, and 9.84%, respectively.

FedReG demonstrates a smaller impact from increased data distribution heterogeneity compared to other algorithms, maintaining the highest global model accuracy and average client accuracy both before and after the decrease in α. A similar trend is observable in the FMNIST dataset, indicating that FedReG possesses higher robustness compared to other algorithms in the face of data heterogeneity challenges.

Table 4: Robustness analysis. Table illustrates the impact of different Dirichlet α values on the accuracy of various algorithms, under the condition of 100 clients and a join rate of 0.2.

Dataset	Cifar10				FMNIST			
α	0.1		0.5		0.1		0.5	
Alg.	G	P	G	P	G	P	G	P
FedAvg	49.50%	49.50%	60.20%	60.20%	81.27%	81.27%	87.61%	87.61%
FedProx	51.32%	51.32%	60.44%	60.44%	80.65%	80.65%	87.47%	87.47%
FedDyn	51.75%	51.75%	57.56%	57.56%	84.08%	84.08%	87.44%	87.44%
Ditto	49.39%	84.68%	60.82%	60.82%	81.63%	96.21%	87.69%	87.69%
FedPer	44.22%	87.93%	53.93%	67.49%	80.06%	97.42%	87.28%	92.01%
FedRep	36.25%	85.91%	43.72%	61.17%	61.43%	96.59%	77.61%	88.22%
FedBABU	39.42%	85.18%	49.65%	64.82%	75.48%	95.34%	83.88%	87.88%
GPFL	30.18%	84.31%	58.39%	73.73%	61.98%	95.67%	85.79%	92.81%
FedRoD	49.85%	88.38%	59.69%	73.94%	80.01%	96.74%	87.32%	93.10%
FedReG	**57.70%**	**89.20%**	**62.15%**	**74.06%**	**88.21%**	**97.58%**	**90.38%**	**94.18%**

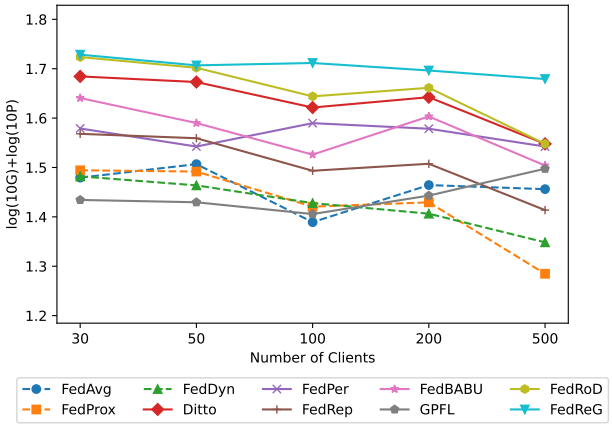

Figure 7: We propose an overall metric $\log(10\times G)+\log(10\times P)$ where G and P are the best *top-1* accuracy of the global model and personal model, respectively.

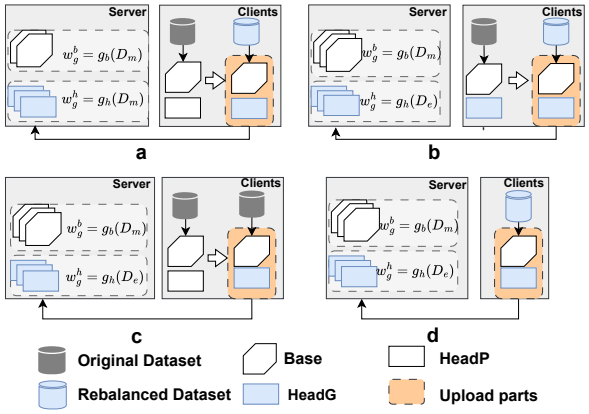

Figure 8: Ablation study.

4.7 Ablation Study

Figure 8 illustrates the testing of four algorithmic variants to assess the global model accuracy under different conditions:

a) Employs the aggregation method of traditional FL, where both the base and the head of the global model are aggregated using the same weight calculation method.
b) Removes the additional head introduced; clients train sequentially on the original dataset and the rebalanced dataset on the same complete base and head model for the aggregation of the global model.
c) Eliminates the rebalanced dataset; the HeadG used for aggregation is trained successively with the original dataset.
d) Removes the step of training the complete model with the original dataset first, using only the rebalanced dataset for training the complete model.

In Figure 8, $g(D)$ is the formula used for calculating the model aggregation weights. The equation is given as:

$$g(D) = \sum_{m=1}^{M} \frac{Dm}{D} w_m. \tag{8}$$

These variants were tested as depicted, focusing on the global model's accuracy. *Variant (a)* aims to demonstrate the effectiveness of our algorithm's modifications to the global model aggregation weights. *Variants (b) and (c)* are designed to confirm the impact of the specially introduced head and the rebalanced dataset for global model aggregation. *Variant (d)* aims to verify the effect of the clients' training framework, which involves sequential training with the original and rebalanced datasets, on the global model.

Data from Table 5 indicates that variant (a) experiences a 2.06% drop in accuracy using the traditional FL aggregation. Since the HeadG, used for global model aggregation, is trained with the rebalanced dataset generated from the users' local original data, it is more sensible to compute the aggregation weight of the global model's head based on the effective sample size.

Table 5: The table presents an ablation study comparison table. The experiments are conducted using the Cifar10 dataset, with a join rate set at 0.2 and the Dirichlet α parameter fixed at 0.1.

FedReG	a	b	c	d	FedAvg
60.50%	58.44%	56.65%	58.23%	55.04%	54.91%

In variant (b), the removal of an additional head, whether it be the client-side HeadP or the aggregation HeadG, shifts the training to involve two datasets training a single complete model. This results in a 3.85% decrease in accuracy, suggesting that sequential training with the original training set and the rebalanced dataset could confuse the model head's understanding of data distribution, reducing the accuracy.

For variant (c), excluding the extra rebalanced dataset results in a 2.27% reduction in accuracy, likely due to the absence of rebalanced dataset fine-tuning for HeadG, which diminishes the aggregated global model's precision.

Finally, variant (d) shows a 5.46% decrease in accuracy after the step of initial training with the original data is removed, yet it still represents a slight improvement over the FedAvg global model. Compared to variant (b), the absence of original data training and relying solely on the rebalanced dataset might weaken the base part of the model's feature learning compared to the performance after training with the original data. Nonetheless, the adjustments in data distribution by the rebalanced dataset still afford a minor enhancement in global model accuracy.

5 TESTBED VALIDATION

For the algorithm discussed earlier, we have also conducted tests on actual devices. The testbed utilized consists of 10 NVIDIA Jetson Nanos[21], serving as clients running the federated learning algorithm, a Lenovo R9000K laptop functioning as the federated learning server and the MQTT message broker[22], and a TP-Link AX3000 router connecting Jetson Nanos and the laptop to form a FL testbed.

The algorithms tested are the same as those listed in the previous tables, and the dataset used for testing is Fashion-MNIST. There are 30 users in total, with each Jetson Nano simulating three clients sequentially. The model used for testing is still ConvNet. The participation rate for each training round is 0.2, meaning 6 out of 30 clients are chosen to participate in each round. The reasons for selecting the Fashion-MNIST dataset and ConvNet are due to their relative simplicity, which does not occupy excessive VRAM and memory on Jetson Nano, allowing for quicker testing speeds.

Accuracy testing for each round is conducted at the start of the round, after the server sends the aggregated model to all clients. The global model, aggregated on the laptop-simulated server, is tested on the server using a complete dataset compiled from all clients' test sets, i.e., the test set of the original dataset. Unlike the simulation phase, client accuracy is first tested using the locally trained model from the previous round.

After testing, clients send the number of correctly predicted samples and the total number of samples to the server. The server then sends the aggregated weights to all clients for updating. Hence, for traditional federated learning algorithms, different global model accuracies and average client accuracies are expected compared to the simulation phase.

Figure 9: Our testbed: 10 Jetson Nano embedded devices connect to an FL server wirelessly via a TP-Link Router.

Table 6: Validation on testbed using the Fmnist dataset with 30 clients, Dirichlet 0.1, and join rate of 0.2.

Dataset	FMNIST	
Joint rate	0.2	
Acc type	Global	Personalized
FedAvg	82.13%	98.05%
FedProx	81.18%	98.05%
FedDyn	83.68%	97.75%
GPFL	48.30%	97.46%
Ditto	81.41%	97.65%
FedPer	72.27%	**98.29%**
FedRep	59.75%	97.88%
FedBABU	78.73%	97.76%
FedRoD	83.96%	98.01%
FedReG	**86.12%**	98.25%

The testbed's communication is implemented via MQTT, with nine topics facilitating the communication process during federated learning.

Table 6 shows the test results generated during the testbed validation process. It can be observed that under the Fashion-MNIST dataset, with 30 clients, our algorithm, FedReG, still aligns with previous conclusions, demonstrating high global model accuracy while maintaining average client accuracy without significant decline. Compared to FedRoD and other PFL algorithms as well as traditional FL algorithms, it achieves higher accuracy, and also shows a slight improvement in average client accuracy over all algorithms except FedPer, ensuring that accuracy is not compromised.

6 CONCLUSION AND FUTURE WORK

In this paper, we have designed FedReG, a personalized learning solution through data decoupling, aiming to address the performance degradation in the global model due to data heterogeneity in federated learning. It has been validated on image data, demonstrating global model performance and average accuracy on par with FedRoD, an algorithm specifically devised to optimize global model accuracy in personalized federated learning. Compared to FedRoD, FedReG exhibits superior global model performance, showcasing enhanced scalability and robustness. Furthermore, when compared to other traditional federated learning algorithms and personalized federated learning algorithms such as GPFL and FedBABU, the additional introduction of the rebalanced dataset and head has not led to a decrease in client average accuracy, but instead, there has been a slight improvement.

The threshold t used in the experiments above was chosen empirically. A future research direction could be to determine a more rational threshold t that reduces local training overhead while maintaining algorithm accuracy is a worthwhile avenue of exploration. Additionally, investigating whether FedReG can be effectively combined with other algorithms that enhance client average accuracy or global model performance is also a promising endeavor worth pursuing.

REFERENCES

[1] 2021. Federated Optimization in Heterogeneous Networks. https://github.com/litian96/FedProx. Last accessed: Nov. 22, 2023.

[2] 2022. Exploiting Shared Representations for Personalized Federated Learning (ICML 2021). https://github.com/lgcollins/FedRep.

[3] 2022. Federated Learning Based on Dynamic Regularization. https://github.com/alpemreacar/FedDyn. Last accessed: Nov. 22, 2023.

[4] 2023. GPFL. https://github.com/TsingZ0/GPFL.

[5] Durmus Alp Emre Acar and et al. 2021. Federated learning based on dynamic regularization. *arXiv preprint arXiv:2111.04263* (2021).

[6] Manoj G. Arivazhagan, Vinay Aggarwal, Aaditya K. Singh, and Sunav Choudhary. 2019. Federated Learning with Personalization Layers. arXiv:1912.00818 [cs.LG]

[7] Hong-You Chen and Wei-Lun Chao. 2022. On Bridging Generic and Personalized Federated Learning for Image Classification. arXiv:2107.00778 [cs.LG]

[8] Yong Cheng, Yang Liu, Tianjian Chen, and Qiang Yang. 2020. Federated learning for privacy-preserving AI. *Commun. ACM* (2020).

[9] Gregory Cohen, Saeed Afshar, Jonathan Tapson, and Andre Van Schaik. 2017. EMNIST: Extending MNIST to handwritten letters. In *IEEE IJCNN*.

[10] Liam Collins, Hamed Hassani, Aryan Mokhtari, and Sanjay Shakkottai. 2021. Exploiting Shared Representations for Personalized Federated Learning. In *Proceedings of the ICML*. https://proceedings.mlr.press/v139/collins21a.html

[11] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. 2019. Autoaugment: Learning augmentation strategies from data. In *Proceedings of the IEEE/CVF CVPR*.

[12] Lydia de la Torre. 2018. A guide to the california consumer privacy act of 2018. *Available at SSRN 3275571* (2018).

[13] Peter Kairouz and et al. 2021. Advances and Open Problems in Federated Learning. *Foundations and Trends® in Machine Learning* 14 (2021). https://doi.org/10.1561/2200000083

[14] Alex Krizhevsky, Geoffrey Hinton, et al. 2009. Learning multiple layers of features from tiny images. (2009).

[15] Viraj Kulkarni, Milind Kulkarni, and Aniruddha Pant. 2020. Survey of personalization techniques for federated learning. In *IEEE WorldS4*.

[16] Tian Li and et al. 2021. Ditto: Fair and robust federated learning through personalization. In *ICML*.

[17] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. 2020. Federated optimization in heterogeneous networks. *Proceedings of Machine learning and systems* (2020).

[18] Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. 2020. On the Convergence of FedAvg on Non-IID Data. arXiv:1907.02189 [stat.ML]

[19] Wei Lim and et al. 2020. Federated learning in mobile edge networks: A comprehensive survey. *IEEE Communications Surveys&Tutorials* (2020).

[20] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *Proceedings of AISTATS*. https://proceedings.mlr.press/v54/mcmahan17a.html

[21] NVIDIA. 2024. Jetson Nano Developer Kit. https://developer.nvidia.com/embedded/jetson-nano-developer-kit.

[22] OASIS Standard. 2015. MQTT Version 3.1.1 Plus Errata 01. https://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html.

[23] Jaehoon Oh, Sangmook Kim, and Se-Young Yun. 2022. FedBABU: Towards Enhanced Representation for Federated Image Classification. arXiv:2106.06042 [cs.LG]

[24] Sebastian Ruder. 2017. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098* (2017).

[25] Canh T Dinh and et al. 2020. Personalized federated learning with moreau envelopes. *Advances in Neural Information Processing Systems* (2020).

[26] Tan, Han Yu, Lizhen Cui, and Qiang Yang. 2022. Towards Personalized Federated Learning. *IEEE Transactions on Neural Networks and Learning Systems* (2022), 1–17. https://doi.org/10.1109/TNNLS.2022.3160699

[27] Paul Voigt and Axel Von dem Bussche. 2017. The eu general data protection regulation (gdpr). *A Practical Guide, 1st Ed., Cham: Springer International Publishing* 10, 3152676 (2017), 10–5555.

[28] Han Xiao, Kashif Rasul, and Roland Vollgraf. 2017. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747* (2017).

[29] Jianqing Zhang and et al. 2023. Gpfl: Simultaneously learning global and personalized feature information for personalized federated learning. In *Proceedings of IEEE/CVF ICCV*.

[30] Yu Zhang and Qiang Yang. 2021. A survey on multi-task learning. *IEEE Transactions on Knowledge and Data Engineering* (2021).

[31] Yue Zhao and et al. 2018. Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582* (2018).

[32] Hangyu Zhu, Jinjin Xu, Shiqing Liu, and Yaochu Jin. 2021. Federated learning on non-IID data: A survey. *Neurocomputing* 465 (2021), 371–390. https://doi.org/10.1016/j.neucom.2021.07.098