8-2024

Is aggregation the only choice? Federated learning via layer-wise model recombination

Ming HU
Singapore Management University, minghu@smu.edu.sg

Zhihao YUE

Xiaofei XIE
Singapore Management University, xfxie@smu.edu.sg

Cheng Chen CHEN

Citation

Is Aggregation the Only Choice? Federated Learning via Layer-wise Model Recombination

Ming Hu
hu.ming.work@gmail.com
Singapore Management University
Singapore, Singapore

Zhihao Yue
51215902034@stu.ecnu.edu.cn
East China Normal University
Shanghai, China

Xiaofei Xie
xfxie@smu.edu.sg
Singapore Management University
Singapore, Singapore

Cheng Chen
chchen@sei.ecnu.edu.cn
East China Normal University
Shanghai, China

Yihao Huang
huangyihao22@gmail.com
Nanyang Technological University
Singapore, Singapore

Xian Wei
xwei@sei.ecnu.edu.cn
East China Normal University
Shanghai, China

Xiang Lian
xlian@kent.edu
Kent State University
Ohio, USA

Yang Liu
yangliu@ntu.edu.sg
Nanyang Technological University
Singapore, Singapore

Mingsong Chen*
mschen@sei.ecnu.edu.cn
East China Normal University
Shanghai, China

Abstract

Although Federated Learning (FL) enables global model training across clients without compromising their raw data, due to the unevenly distributed data among clients, existing Federated Averaging (FedAvg)-based methods suffer from the problem of low inference performance. Specifically, different data distributions among clients lead to various optimization directions of local models. Aggregating local models usually results in a low-generalized global model, which performs worse on most of the clients. To address the above issue, inspired by the observation from a geometric perspective that a well-generalized solution is located in a flat area rather than a sharp area, we propose a novel and heuristic FL paradigm named FedMR (Federated Model Recombination). The goal of FedMR is to guide the recombined models to be trained towards a flat area. Unlike conventional FedAvg-based methods, in FedMR, the cloud server recombines collected local models by shuffling each layer of them to generate multiple recombined models for local training on clients rather than an aggregated global model. Since the area of the flat area is larger than the sharp area, when local models are located in different areas, recombined models have a higher probability of locating in a flat area. When all recombined models are located in the same flat area, they are optimized towards the same direction. We theoretically analyze the convergence of model recombination. Experimental results show that, compared with state-of-the-art FL methods, FedMR can significantly improve the inference accuracy without exposing the privacy of each client.

*Corresponding author.

CCS Concepts

• **Computing methodologies** → *Distributed artificial intelligence.*

Keywords

Federated Learning, Model Recombination, Non-IID, Generalization

1 Introduction

Federated Learning (FL) [25, 32, 40] has been widely acknowledged as a promising means to design large-scale distributed Artificial Intelligence (AI) applications, e.g., Artificial Intelligence of Things (AIoT) systems [13, 37, 47], healthcare systems [2, 46], and recommender systems [35, 42]. Unlike conventional Deep Learning (DL) methods, the cloud-client architecture based FL supports the collaborative training of a global DL model among clients without compromising their raw data [3]. In each FL training round, the cloud server first dispatches the global model to its selected clients for local training and then gathers the corresponding gradients of trained models from clients for aggregation. In this way, clients can train a global model without sharing data.

Although FL enables effective collaborative training among multiple clients while protecting data privacy, existing FL methods suffer from the problem of "weight divergence" [21]. Especially when the data on the clients are non-IID (Identically and Independently Distributed) [1, 38], the optimal directions of local models on clients and the aggregated global model on the cloud server are significantly inconsistent, resulting in serious inference performance degradation of the global model. To improve FL performance in non-IID scenarios, various FL methods have been studied, e.g., client grouping-based methods [45], global control

variable-based methods [17, 22, 28], Knowledge Distillation (KD)-based methods[30, 39, 48], and mutation-based methods [14]. The basic ideas of these solutions are to guide the local training on clients [17, 22] or adjust parameter weights for model aggregation [30, 48].

Although these methods are promising in alleviating the impact of data heterogeneity, most of them adopt the well-known Federated Averaging (FedAvg)-based paradigm, which may potentially reduce generalization performance. This is mainly because FedAvg paradigm only aggregates the parameters of collected local models and initializes local training by clients with the same global models. Specifically, since the data distribution among clients is different, the optimal directions of the local models are diverse. On the one hand, although the aggregation operation can achieve knowledge sharing among multiple local models, it can still neglect the specific knowledge learned by local models, which seriously limits the inference performance of the global model. On the other hand, since FedAvg only uses the same global model for local training, FL training inevitably results in notorious *stuck-at-local-search* problems during local training. As a result, the global model based on simple statistical averaging cannot accurately reflect both individual efforts and the potential of local models in the search for optimal global models. Therefore, *how to overcome the shortcomings of the FedAvg-based paradigm and improve the performance of FL in non-IID scenarios is an important challenge.*

Some recent research on model training indicates that, from the perspective of the loss landscapes of DL models, optimal solutions with well generalization performance often lie in flat valleys, while the inferior ones are always located in sharp ravines [11, 24]. Inspired by the above observation, to collaboratively train a well-generalized model, FL needs to guide the local training towards a more flat area. Since the direction of gradient descent is stochastic, compared to using the same global model, using multiple global models for local training has a greater probability that the existing model can optimize to a flatter area. Since flat areas are usually larger than sharp areas, intuitively, the exchange of the corresponding parameters among multiple models rather than aggregation can allow them to be displaced in the solution space. When a model is stuck in a sharp area, the parameter exchange may make it escape from the sharp area. With continuous training and parameter exchange, when multiple models are located in the same flat area, these models will optimize in the same direction, that is, the center of the flat area.

Inspired by the above intuition, this paper proposes a novel FL paradigm called FedMR (**Fed**erated **M**odel **R**ecombination), which can effectively help the training of local models escape from sharp area. Unlike FedAvg that aggregates all the collected local models in each FL training round, FedMR randomly shuffles the parameters of different local models within the same layers, and recombines them to form new local models. In this way, FedMR can derive diversified models that can effectively escape local optimal solutions for the local training of clients. The main contributions of this paper can be summarized as follows:

- We propose a novel FL paradigm named FedMR, which contains a newly layer-wise model recombination method to replace the traditional FedAvg-based model aggregation with the aim of improving FL inference performance.

- We introduce a two-stage training scheme for FedMR, which combines the merits of both model recombination and aggregation to accelerate the overall FL training process.
- We theoretically prove the convergence of FedMR in convex scenarios and conduct empirical experiments to validate the convergence of FedMR in non-convex scenarios.
- We conduct extensive experiments on various well-known models and datasets to show both the effectiveness and compatibility of FedMR.

2 Related Work

To address the problem of uneven data distributions, exiting solutions can be mainly classified into four categories, i.e., client grouping-based methods, global control variable-based methods, knowledge distillation-based methods, and mutation-based methods. The *device grouping-based methods* group and select clients for aggregation based on the data similarity between clients. For example, FedCluster [8] divides clients into multiple clusters and performs multiple cycles of meta-update to boost the overall FL convergence. Based on either sample size or model similarity, CluSamp [9] groups clients to achieve a better client representativity and a reduced variance of client stochastic aggregation parameters in FL. By modifying the penalty terms of loss functions during FL training, the *global control variable-based methods* can be used to smooth the FL convergence process. For example, FedProx [28] regularizes local loss functions with the squared distance between local models and the global model to stabilize the model convergence. Similarly, SCAFFOLD [22] uses global control variables to correct the "client-drift" problem in the local training process. *Knowledge Distillation (KD)-based methods* adopt soft targets generated by the "teacher model" to guide the training of "student models". For example, by leveraging a proxy dataset, Zhu et al. [48] proposed a data-free knowledge distillation method named FedGen to address the heterogeneous FL problem using a built-in generator. With ensemble distillation, FedDF [30] accelerates the FL training by training the global model through unlabeled data on the outputs of local models. *Mutation-based methods* attempt to mutate the global model to generate multiple mutated intermediate models for local training. For example, FedMut [14] utilizes the gradients to mutate the global model and dispatches the mutated models for local training.

Although the above methods can optimize FL performance from different perspectives, since coarse-grained model aggregation is performed, the inference capabilities of local models are still strongly restricted. Furthermore, most of them cannot avoid non-negligible communication and computation overheads or the risk of data privacy exposure. In addition, many FL methods have been proposed to address device heterogeneity problems. To effectively train on devices with different hardware resources, some methods [4, 18, 20, 41] utilize heterogeneous models for local training. To avoid stragglers caused by uneven computing capability or uncertainty [15], existing methods [16, 44] attempt to perform a wise client scheduling to achieve asynchronous FL training. Note that this paper only focuses on the data heterogeneity problem.

To the best of our knowledge, FedMR is the first attempt using model recombination rather than aggregation for FL. Since FedMR considers the specific characteristics and efforts of local models, it

Is Aggregation the Only Choice? Federated Learning via Layer-wise Model Recombination

KDD '24, August 25–29, 2024, Barcelona, Spain.

can further mitigate the weight divergence problem, thus achieving better inference performance than state-of-the-art FL methods.

3 Motivation

3.1 Intuition

Comparison between FedAvg and Independent Training. Figure 1 illustrates the FL training processes on the same loss landscape using FedAvg and Independent training (Indep), respectively, where the server of Indep just shuffles the received local models and then randomly dispatches them to clients without aggregation. In each subfigure, the local optima are located within the areas surrounded by solid red lines. Note that since the upper surrounded area is flatter than the lower surrounded area in the loss landscape, the solutions within it will exhibit better generalization.
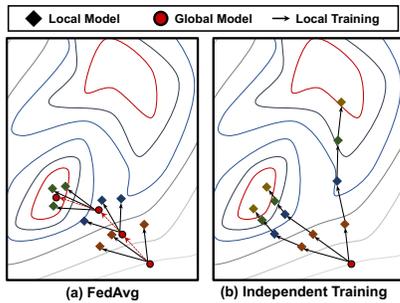
Figure 1: Training processes on the same loss landscape.

As shown in Figure 1(a), along with the training process, the aggregated global models denoted by red circles gradually move toward the lower sharp area with inferior solutions, though the optimizations of some local models head toward the upper surrounded area with better solutions. Such biased training is mainly because the local training starts from the same global model in each FL round. As an alternative, due to the lack of aggregation operation, the local models of Indep may converge in different directions as shown in Figure 1(b). In this case, even if some local training in Idep achieves a better solution than the one obtained by FedAvg, due to the diversified optimization directions of local models, such an important finding can be eclipsed by the results of other local models. Clearly, there is a lack of mechanisms for Indep that can guide the overall training toward such superior solutions.

Intuition of Model Recombination. Based on the Indep training results shown in Figure 1(b), Figure 2 illustrates the intuition of our model recombination method, where the FL training starts from the three local models (denoted by yellow diamonds in round 1) obtained in figure 1(b). At the beginning of round 1, two of the three local models are located in the sharp ravine. In other words, without model recombination, the training of such two local models may get stuck in the lower surrounded area. However, due to the weight adjustment by shuffling the layers among local models, we can find that the three recombined models (denoted by yellow squares) are sparsely scattered in the loss landscape, which enables the local training escape from local optima. According to [10, 43], a small perturbation of the model weights can make it easier for
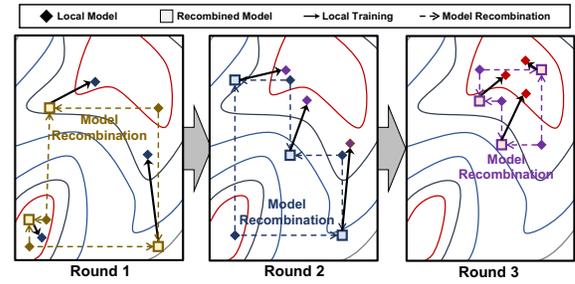
Figure 2: An example of model recombination.

local training to jump out of sharp ravines rather than flat valleys. In other words, the recombined models are more likely to converge toward flat valleys along the local training. For example, in the end of round 3, we can find that all three local models are located in the upper surrounded area, where their aggregated model has better generalization performance than the one achieved in Figure 1(a).
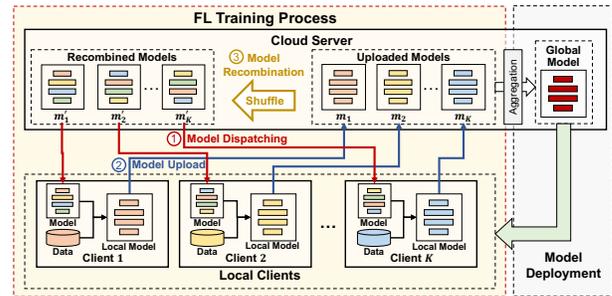
Figure 3: Our FedMR approach

4 FedMR Approach

4.1 Overview of FedMR

Figure 3 presents the framework and workflow of FedMR, which consists of two process, i.e., FL training process and the model deployment process. In FL training process, unlike FedAvg-based methods that use the same global model for local training, FedMR adopts multiple homogeneous intermediate models for local training, where each client model is dispatched one model. Specifically, each training round involves three specific steps as follows:

- **Step 1 (Model Dispatching):** The cloud server dispatches K intermediate models to K selected clients, respectively, according to their indices, where K denotes the number of activated clients participating in each FL training round. Note that in FedMR different clients will receive different models for the local training.
- **Step 2 (Model Upload):** Once its local training is finished, a client needs to upload the parameters of its newly updated local model to the cloud server.
- **Step 3 (Model Recombination):** The cloud server decomposes received local models into multiple layers individually in the same manner and conducts the random shuffling of

Algorithm 1 Implementation of FedMR

Input: i) rnd, # of training rounds; ii) S_c, the set of clients; iii) K, # of clients participating in each FL round.
Output: w^{glb}, the parameters of trained global model.
FedMR(rnd, S_{dev}, K)

1: $L_m \leftarrow [w_1^1, w_1^2, ..., w_1^K]$ // initialize model list
2: **for** r = 1, ..., rnd **do**
3:   $L_r \leftarrow$ Random select K clients from S_c
    /*parallel for block*/
4:   **for** i = 1, ..., K **do**
5:     $v_{r+1}^i \leftarrow ClientUpdate(L_m[i], L_r[i])$
6:     $L_m[i] \leftarrow v_{r+1}^i$
7:   **end for**
8:   $[w_{r+1}^1, w_{r+1}^2, ..., w_{r+1}^K] \leftarrow ModelRcombine(L_m)$
9:   $L_m \leftarrow [w_{r+1}^1, w_{r+1}^2, ..., w_{r+1}^K]$
10: **end for**
11: $w^{glb} \leftarrow \frac{1}{K} \sum_{i=1}^K w_{rnd+1}^i$
12: **return** w^{glb}

the same layers among different local models. Then, by concatenating layers from different sources in order, a new local model can be reconstructed. Note that any decomposed layer of the uploaded model will eventually be used by one and only one of the recombined models.

In the model deployment process, the cloud server aggregates all the intermediate models to generate a global model and deploys the global model to all the local clients for specific tasks.

4.2 Implementation of FedMR

Algorithm 1 details the implementation of FedMR. Line 1 initializes the model list L_m, which includes K initial models. Lines 2-10 performs rnd rounds of FedMR training. In each round, Line 3 selects K random clients to participate in the model training and creates a client list L_r. Lines 4-7 conduct the local training on clients in parallel, where Line 5 applies the local model $L_m[i]$ on client $L_r[i]$ for local training by using the function *ClientUpdate*, and Line 6 achieves a new local model after the local training. After the cloud server receives all the K local models, Line 8 uses the function *ModelRcombine* to recombine local models and generate K new local models, which are saved in L_m as shown in Line 9. Finally, Lines 11-12 will report an optimal global model that is generated based on L_m. Note that the cloud server will dispatch the global model to all the clients for the purpose of inference rather than local training. The following parts will detail the key components of FedMR. Since FedMR cannot adapt to the secure aggregation mechanism [5], to further protect privacy, we present an extended secure recombination mechanism in Appendix B, which enables the exchange of partial layers among clients before local training or model uploading to ensure that the cloud server cannot directly obtain the gradients of each local model.

4.2.1 Local Model Training. Unlike conventional FL methods that conduct local training on clients starting from the same aggregated model, in each training round FedMR uses different recombined models (i.e., K models in the model list L_m) for the local training

(a) Example of Model Aggregation

(b) Example of Model Recombination
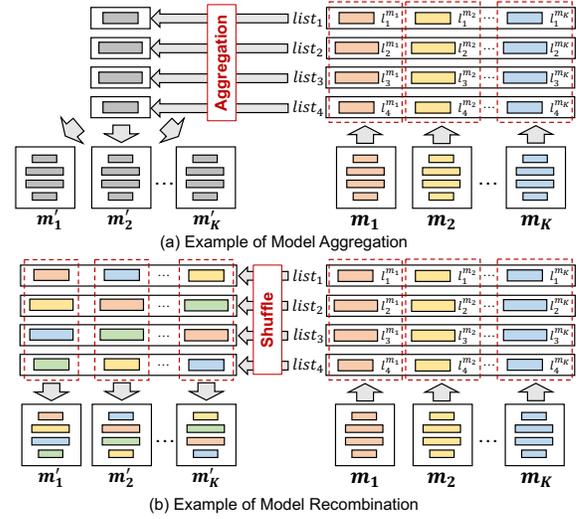
Figure 4: Example of model aggregation and recombination

purpose. Note that, in the whole training phase, FedMR only uses K ($K \leq |S_c|$) models, since there are only K devices activated in each training round. Let w_r^c be the parameters of some model that is dispatched to the cth client in the rth training round. In the rth training round, we dispatch the ith model in L_m to its corresponding client using $w_r^{L_r[i]} = L_m[i]$. Based on the recombined model, FedMR conducts the local training on client $L_r[i]$ as follows:

$$v_{r+1}^{L_r[i]} = w_r^{L_r[i]} - \eta \nabla f_{L_r[i]}(w_r^{L_r[i]}),$$

$$s.t., \quad f_{L_r[i]}(w_r^{L_r[i]}) = \frac{1}{|D_{L_r[i]}|} \sum_{j=1}^{|D_{L_r[i]}|} \ell(w_r^{L_r[i]}; x_j; y_j),$$

where $v_r^{L_r[i]}$ indicates parameters of the trained local model, $D_{L_r[i]}$ denotes the dataset of client $L_r[i]$, η is the learning rate, $\ell(\cdot)$ is the loss function, x_j is the jth sample in $D_{L_r[i]}$, and y_j is the label of x_j. Once the local training is finished, the client needs to upload the parameters of its trained local model to the cloud server by updating L_m using $L_m[i] = v_{r+1}^{L_r[i]}$. Similar to traditional FL methods, in each training round, FedMR needs to transmit the parameters of $2K$ models between the cloud server and its selected clients.

4.2.2 Model Recombination. Typically, a DL model consists of multiple layers, e.g., convolutional layers, pooling layers, and Fully Connected (FC) layers. To simplify the description of our model recombination method, we do not explicitly present the layer types here. Let $w_x = \{l_1^x, l_2^x, ..., l_n^x\}$ be the parameters of model x, where l_i^x ($i \in [n]$) denotes the parameters of the ith layer of model x.

In each FL round, FedMR needs to perform model recombination based on L_m to obtain new models for local training. Figure 4 presents an example of model aggregation and model recombination. When clients receive all the trained local models (i.e., m_1, m_2, ..., m_K), the cloud server needs to decouple the layers of these models individually. Note that since all the local models are homogeneous, the corresponding layers of the local models have the

Is Aggregation the Only Choice? Federated Learning via Layer-wise Model Recombination

KDD '24, August 25–29, 2024, Barcelona, Spain.

same structure. For example, the model m_1 can be decomposed into four layers. Assuming that the local models are with an architecture of w, to enable recombination, FedMR then constructs n lists, where the kth ($k \in [n]$) list contains all the kth layers of the models in L_m. As an example shown in Figure 4(b), FedMR constructs four lists (i.e., $list_1$-$list_4$) for the K models (i.e., m_1-m_K), where each list consists of K elements (i.e., K layers with the same index). After shuffling each list, FedMR generates $|L_m|$ recombined models based on shuffled results. For example, the top three layers of the recombined model m_1' come from the models m_1, m_2 and m_K, respectively. For comparison, Figure 4(a) presents an example of model aggregation, which aggregates the layers of each list to generate an aggregated model. The aggregation-based methods dispatch the aggregated model to K clients, while FedMR dispatches a different recombined model to each client.

4.3 Two-Stage Training Scheme for FedMR

Although FedMR enables finer FL training, when starting from blank models, FedMR converges more slowly than traditional FL methods at the beginning. This is mainly because, due to the low matching degree between layers in the recombined models, the model recombination operation in this stage requires more local training time to re-construct the new dependencies between layers. To accelerate the overall convergence, we propose a two-stage training scheme for FedMR, consisting of the *aggregation-based pre-training stage* and *model recombination stage*. In the first stage, we train the local models coarsely using the FedAvg-based aggregation, which can quickly form a pre-trained global model. In the second stage, from the pre-trained models, FedMR dispatches recombined models to clients for local training. Due to the synergy of both FL paradigms, the overall FedMR training time can be reduced.

4.4 Convergence Analysis

Based on the assumptions posed on the loss functions of local clients in FedAvg [29], this subsection performs the convergence analysis for FedMR.

Assumption 4.1. For $i \in \{1, 2, \cdots, K\}$, f_i is L-smooth satisfying $||\nabla f_i(x) - \nabla f_i(y)|| \le L||x - y||$.

Assumption 4.2. For $i \in \{1, 2, \cdots, K\}$, f_i is μ-strongly convex satisfying $f(x) \ge f(y) + (x - y)^T \nabla f(y) + \frac{\mu}{2}||x - y||^2$, where $\mu \ge 0$.

Assumption 4.3. The variance of stochastic gradients is upper bounded by θ^2, and the expectation of squared norm of stochastic gradients is upper bounded by G^2, i.e., $\mathbb{E}||\nabla f_k(w; \xi) - \nabla f_k(w)||^2 \le \theta^2$, $\mathbb{E}||\nabla f_k(w; \xi)||^2 \le G^2$, where ξ is a data batch of the kth client in the tth FL round.

Based on the implementation of function *ModelRecombine(·)*, we derive the following two lemmas for the model recombination:

Lemma 4.4. *Assume that in FedMR there are K clients participating in every FL training round. Let $\{v_r^1, v_r^2, .., v_r^K\}$ and $\{w_r^1, w_r^2, .., w_r^K\}$ be the set of trained local model weights and the set of recombined model weights generated in the $(r-1)^{th}$ round, respectively. Assume x is a vector with the same size as that of v_r^k. We have*

$$\sum_{k=1}^{K} v_r^k = \sum_{k=1}^{K} w_r^k, \text{ and } \sum_{k=1}^{K} ||v_r^k - x||^2 = \sum_{k=1}^{K} ||w_r^k - x||^2.$$

We prove Theorem 1 based on Lemmas 4.4. Please refer to Appendix A for the proof. Note that different from FedAvg, Lemmas 4.4 is the key lemma for the proof of FedMR.

Theorem 1. *(Convergence of FedMR) Let Assumption 4.1, 4.2, and 4.3 hold. Assume that E is the number of SGD iterations conducted within one FL round, model recombination is conducted at the end of each FL round, and the whole training terminates after n FL rounds. Let $T = n \times E$ be the total number of SGD iterations conducted so far, and $\eta_k = \frac{2}{\mu(T+\gamma)}$ be the learning rate. We can have*

$$\mathbb{E}[f(\overline{w}_T)] - f^\star \le \frac{L}{2\mu(T+\gamma)} \Big[\frac{4B}{\mu} + \frac{\mu(\gamma+1)}{2}\Delta_1 \Big],$$

where $B = 10L\Gamma + 4(E-1)^2 G^2$, $\overline{w}_T = \sum_{k=1}^{K} w_T^k$.

Theorem 1 indicates that the difference between the current loss $f(\overline{w}_T)$ and the optimal loss $f^\star$ is inversely related to t. From Theorem 1, we can find that the convergence rate of FedMR is similar to that of FedAvg, which has been analyzed in [29].

5 Experimental Results

5.1 Experimental Settings

To evaluate the effectiveness of FedMR, we implemented FedMR on top of a cloud-based architecture. Since it is impractical to allow all the clients to get involved in the training processes simultaneously, we assumed that there are only 10% of clients participating in the local training in each FL round. To enable fair comparison, all the investigated FL methods including FedMR set SGD optimizer with a learning rate of 0.01 and a momentum of 0.9. For each client, we set the batch size of local training to 50, and performed five epochs for each local training. All the experimental results were obtained from an Ubuntu workstation with Intel i9 CPU, 32GB memory, and NVIDIA RTX 3080 GPU.

Baseline Method Settings. We compared the test accuracy of FedMR with seven baseline methods, i.e., FedAvg [32], FedProx [28], FedGen [48], CluSamp [9], FedExP [19], FedASAM [6], and Fed-Mut [14]. Here, FedAvg is the most classical FL method, while the other five methods are state-of-the-art (SOTA) representations of the four kinds of FL optimization methods introduced in the related work section. Specifically, FedProx, FedExP, and FedASAM are global control variable-based methods, FedGen is a KD-based approach, CluSamp is a device grouping-based method, and FedMut is a mutation-based method. For FedProx, we used a hyper-parameter μ to control the weight of its proximal term, where the best values of μ for CIFAR-10, CIFAR-100, and FEMNISTvare 0.01, 0.001, and 0.1, respectively. For FedGen, we adopted the same server settings in [48]. For CluSamp, the clients were clustered based on the model gradient similarity described in [9].

Dataset Settings. We investigated the performance of our approach on three well-known datasets, i.e., CIFAR-10, CIFAR-100 [23], and FMNIST [7]. We adopted the Dirichlet distribution [12] to control the heterogeneity of client data for both CIFAR-10 and CIFAR-100. Here, the notation $Dir(\alpha)$ indicates a different Dirichlet distribution controlled by α, where a smaller α means higher data heterogeneity of clients. Note that, different from datasets CIFAR-10 and CIFAR-100, the raw data of FEMNIST are naturally

non-IID distributed. Since FEMNIST takes various kinds of data heterogeneity into account, we did not apply the Dirichlet distribution on FEMNIST. For both CIFAR-10 and CIFAR-100, we assumed that there are 100 clients in total participating in FL. For FEMNIST, we only considered one non-IID scenario involving 180 clients, where each client hosts more than 100 local data samples.

Model Settings. To demonstrate the pervasiveness of our approach, we developed different FedMR implementations based on three different models (i.e., CNN, ResNet-20, VGG-16). Here, we obtained the CNN model from [32], which consists of two convolutional layers and two FC layers. When conducting FedMR based on the CNN model, we directly applied the model recombination for local training on it without pre-training a global model, since CNN here only has four layers. We obtained both ResNet-20 and VGG-16 models from Torchvision [36]. · When performing FedMR based on ResNet-20 and VGG-16, due to the deep structure of both models, we adopted the two-stage training scheme, where the first stage lasts for 100 rounds to obtain a pre-trained global model.

5.2 Validation for Intuitions

Independent Training. Based on the settings presented in Section 5.1, we conducted the experiments to evaluate the effectiveness of each local model in Indep. The FL training is based on the ResNet-20 model and dataset CIFAR-10, where we set $\alpha = 0.5$ for non-IID scenarios. Figures 5 compares Indep with FedAvg from the perspectives of both test loss and inference accuracy. Due to the space limitation, for Indep here we only present the results of its four random local models (denoted by Model-1, Model-2, Model-3, and Model-4). To enable a fair comparison with FedAvg, although there is no aggregated global model in Indep, we considered the aggregated model of all its local models for each FL round, whose results are indicated by the notion "IndepAggr". From Figure 5, we can find all the local models in Indep can achieve higher accuracy and lower loss than those of FedAvg, though their loss and accuracy curves fluctuate more sharply. Moreover, IndepAggr exhibits much worse performance than the other references. This is mainly because, according to the definition of Indep, each local model needs to traverse multiple clients along with the FL training processes, where the optimization directions of client models differ in the corresponding loss landscape.

(a) Test Loss   (b) Accuracy
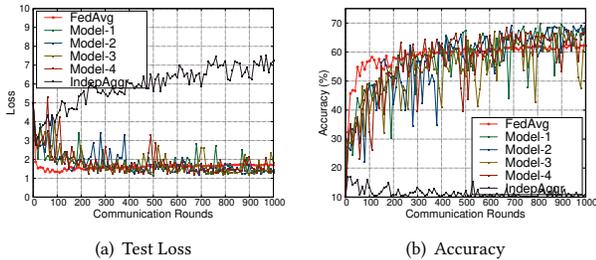
Figure 5: FedAvg vs. Indep.

Model Recombination. To validate the intuition about the impacts of model recombination as presented in Section 3, we conducted three experiments on CIFAR-10 dataset using ResNet-20

model. Our goal is to figure out the following three questions: i) by using model recombination, can all the models eventually have the same optimization direction; ii) compared with FedAvg, can the global model of FedMR eventually converge into a more flat solution; and iii) can the global model of FedMR eventually converge to a more generalized solution?
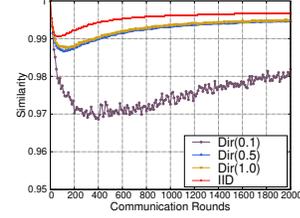
Figure 6: Cosine similarity of local models in FedMR.

Figure 6 presents the average cosine similarity between all the intermediate models, taking four different client data distributions into account. We can observe that the model similarity decreases first and gradually increases in all the investigated IID and non-IID scenarios. Due to the nature of Stochastic Gradient Descent (SGD) mechanism and the data heterogeneity among clients, all local models are optimized toward different directions at the beginning of training. However, as the training progresses, most local models will be located in the same flat valleys, leading to similar optimization directions for local models. These results are consistent with our intuition as shown in Figure 2.

(a) FedAvg w/ $\alpha = 0.1$ (b) FedMR w/ $\alpha = 0.1$

(c) FedAvg w/ $\alpha = 1.0$ (d) FedMR w/ $\alpha = 1.0$
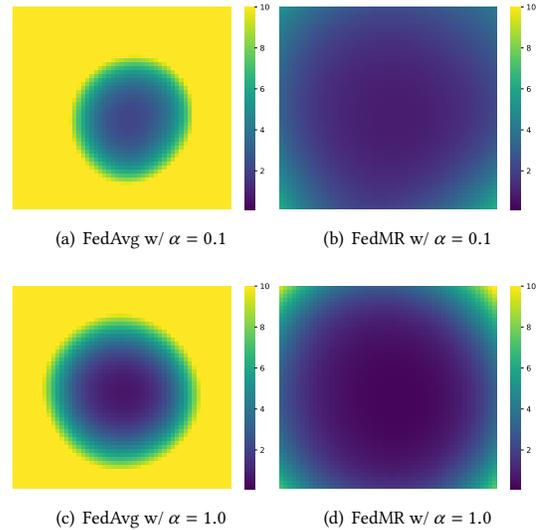
Figure 7: Comparison of loss landscapes with different FL and client data settings.

Figure 7 compares the loss landscapes of final global models obtained by FedAvg and FedMR with different client data settings, respectively. We can find that, compared with FedMR counterparts,

Is Aggregation the Only Choice? Federated Learning via Layer-wise Model Recombination

KDD '24, August 25–29, 2024, Barcelona, Spain.

(a) $\alpha = 0.1$

(b) $\alpha = 0.5$

(c) $\alpha = 1.0$

(d) IID
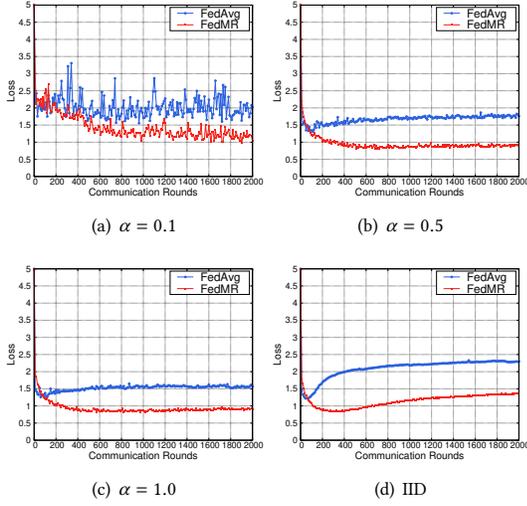
Figure 8: Comparison of test losses of FedAvg and FedMR with different client data settings.

the global models trained by FedAvg are located in sharper solutions, indicating the generalization superiority of final global models achieved by FedMR.

Figure 8 compares the test losses for the global models of FedAvg and FedMR (without using two-stage training) within different IID and non-IID scenarios. Note that here, the global models of FedMR are only for the purpose of fair comparison rather than local model initialization. Due to the superiority in generalization, we can observe that the models trained by FedMR outperform those by FedAvg for all four cases.

5.3 Performance Comparison

We compared the performance of our FedMR approach with seven SOTA baselines. For datasets CIFAR-10 and CIFAR-100, we considered both IID and non-IID scenarios (with $\alpha = 0.1, 0.5, 1.0$, respectively). For dataset FEMNIST, we considered its original non-IID settings [7].

5.3.1 Comparison of Test Accuracy. Table 1 compares FedMR with the SOTA FL methods, considering both non-IID and IID scenarios based on three different DL models. The first two columns denote the model type and dataset type, respectively. Note that to enable fair comparison, we cluster the test accuracy results generated by the FL methods based on the same type of local models. The third column shows different distribution settings for client data, indicating the data heterogeneity of clients. The fourth column has eight sub-columns, which present the test accuracy information together with its standard deviation for all the investigated FL methods, respectively.

From Table 1, we can observe that FedMR can achieve the highest test accuracy in all the scenarios regardless of model type, dataset type, and data heterogeneity. For CIFAR-10 and CIFAR-100, we can find that FedMR outperforms the seven baseline methods significantly in both non-IID and IID scenarios. For example, when dealing

with a non-IID CIFAR-10 scenario ($\alpha = 0.1$) using ResNet-20-based models, FedMR achieves test accuracy with an average of 58.40%, while the second highest average test accuracy obtained by FedMut is only 50.75%. Note that the performance of FedMR on FEMNIST is not as notable as the one on both CIFAR-10 and CIFAR-100. This is mainly because the classification task on FEMNIST is much simpler than the ones applied on datasets CIFAR-10 and CIFAR-100, which leads to the high test accuracy of the seven baseline methods. However, even in this case, FedMR can still achieve the best test accuracy among all the investigated FL methods.

5.3.2 Comparison of Model Convergence. Figure 9 presents the convergence trends of the seven FL methods (including FedMR) on the CIFAR-100 dataset. Note that here the training of FedMR is based on our proposed two-stage training scheme, where the first stage uses 100 FL training rounds to achieve a pre-trained model. Here, to enable fair comparison, the test accuracy of FedMR at some FL training rounds is calculated by an intermediate global model, which is an aggregated version of all the local models within that round. The four sub-figures show the results for different data distributions of clients. This figure shows that FedMR outperforms the other six FL methods consistently in both non-IID and IID scenarios. This is mainly because FedMR can easily escape from the stuck-at-local-search due to the model recombination operation in each FL round. Moreover, due to the fine-grained training, we can observe that the learning curves in each sub-figure are much smoother than the ones of other FL methods. We also compared CNN- and VGG-16-based FL methods and found similar observations.

5.4 Ablation Study

5.4.1 Impacts of Activated Clients . Figure 10 compares the learning trends between FedMR and six baselines for a non-IID scenario ($\alpha = 0.1$) with both ResNet-20 model and CIFAR-10 dataset, where the numbers of activated clients are 5, 10, 20, 50, and 100, respectively. From Figure 10, we can observe that FedMR achieves the best inference performance for all cases. We can also observe that too few activated clients (i.e., $K = 5$) result in a degradation of the accuracy of the global model in all the FL methods. We find that when the number of activated clients increases, the convergence fluctuations reduce significantly and FedMR achieves the smallest fluctuations compared to all the baselines for all cases.

5.4.2 Impacts of Model Layer Partitioning. To show the effectiveness of our layer-wise model recombination scheme, we evaluated the FedMR performance using different model layer partitioning strategies. We use "FedMR-px" to denote that the model is divided into $\lceil \frac{1}{x} \rceil$ ($x \in (0, 1.0]$) segments, where the model recombination is based on segments rather than layers. Note that FedMR does not divide a single layer into multiple segments. Instead, in FedMR each segment involves multiple layers (i.e., one or more layers), and the FedMR takes the recombination over segments among different clients. Specifically, for the jth layer, it belongs to $\lceil \frac{j}{x \times N} \rceil^{th}$ segment. Since $x \in (0, 1.0]$, each segment contains at least one layer. Note that $x = 1.0$ indicates an extreme case, where local models are randomly dispatched to clients without recombination.

Figure 11 presents the ablation study results on CIFAR-10 dataset using ResNet-20-based and VGG-16-based FedMR, where the data

Table 1: Test accuracy comparison for both non-IID and IID scenarios using three DL models

Model	Datas.	Heter. Set.	Test Accuracy (%)							
			FedAvg	FedProx	FedGen	CluSamp	FedExP	FedASAM	FedMut	**FedMR**
CNN	Cifar-10	0.1	50.66 ± 1.18	50.61 ± 1.21	50.27 ± 2.03	50.24 ± 1.04	53.20 ± 1.33	48.83 ± 1.86	<u>52.88 ± 0.59</u>	**54.63 ± 0.55**
		0.5	54.42 ± 0.76	54.28 ± 1.29	53.66 ± 0.68	55.07 ± 0.93	55.12 ± 0.57	52.02 ± 0.43	<u>57.29 ± 0.61</u>	**59.81 ± 0.60**
		1.0	57.03 ± 0.62	56.44 ± 0.68	56.19 ± 0.54	56.16 ± 0.56	57.12 ± 0.34	55.08 ± 0.61	<u>58.88 ± 0.47</u>	**60.77 ± 0.48**
		IID	57.21 ± 0.27	57.00 ± 0.11	57.35 ± 0.19	58.13 ± 0.35	57.05 ± 0.15	54.60 ± 0.11	<u>58.81 ± 0.21</u>	**63.74 ± 0.18**
	Cifar-100	0.1	29.12 ± 0.52	29.44 ± 0.73	26.14 ± 0.92	28.51 ± 1.14	30.43 ± 0.39	28.95 ± 0.48	<u>31.23 ± 0.35</u>	**34.47 ± 0.54**
		0.5	32.41 ± 0.77	32.48 ± 0.81	29.19 ± 0.67	32.63 ± 0.60	33.12 ± 0.51	32.06 ± 0.98	<u>34.46 ± 0.69</u>	**37.41 ± 0.30**
		1.0	32.66 ± 0.51	33.10 ± 0.41	29.94 ± 0.51	32.65 ± 0.48	33.32 ± 0.38	32.45 ± 0.61	<u>35.12 ± 0.42</u>	**39.15 ± 0.30**
		IID	32.75 ± 0.20	32.57 ± 0.21	30.95 ± 0.32	32.77 ± 0.11	32.48 ± 0.14	32.35 ± 0.29	<u>34.49 ± 0.23</u>	**40.64 ± 0.17**
	FEMNIST	−	82.97 ± 0.37	83.15 ± 0.41	82.35 ± 0.40	82.31 ± 0.32	83.28 ± 0.29	83.49 ± 0.27	<u>83.62 ± 0.33</u>	**83.76 ± 0.24**
ResNet-20	Cifar-10	0.1	42.14 ± 3.91	43.25 ± 3.18	44.19 ± 2.42	41.64 ± 2.04	45.18 ± 2.43	45.22 ± 4.06	<u>50.75 ± 1.85</u>	**59.20 ± 1.22**
		0.5	58.70 ± 0.86	59.33 ± 0.77	60.64 ± 0.83	58.74 ± 0.82	59.74 ± 0.92	63.49 ± 1.10	<u>63.34 ± 0.70</u>	**72.41 ± 0.17**
		1.0	64.33 ± 0.25	64.75 ± 0.33	64.41 ± 0.29	63.42 ± 0.45	64.48 ± 0.31	67.62 ± 0.41	<u>68.09 ± 0.25</u>	**75.16 ± 0.31**
		IID	65.72 ± 0.22	65.95 ± 0.23	66.31 ± 0.23	65.36 ± 0.18	65.58 ± 0.24	69.65 ± 0.10	<u>69.77 ± 0.19</u>	**77.48 ± 0.10**
	Cifar-100	0.1	34.22 ± 1.01	34.52 ± 0.68	35.76 ± 1.11	33.23 ± 0.78	36.09 ± 0.71	37.31 ± 0.66	<u>38.79 ± 0.55</u>	**44.61 ± 1.48**
		0.5	42.16 ± 0.43	41.37 ± 0.49	45.03 ± 0.96	41.54 ± 0.52	41.96 ± 0.56	44.29 ± 0.52	<u>46.55 ± 0.55</u>	**54.26 ± 0.45**
		1.0	43.32 ± 0.38	43.00 ± 0.45	46.60 ± 0.39	43.63 ± 0.39	43.68 ± 0.23	46.74 ± 0.31	<u>48.41 ± 0.25</u>	**55.72 ± 0.36**
		IID	45.14 ± 0.28	45.40 ± 0.27	48.33 ± 0.25	44.76 ± 0.24	45.04 ± 0.24	48.59 ± 0.20	<u>48.65 ± 0.17</u>	**59.24 ± 0.32**
	FEMNIST	−	79.09 ± 0.54	78.89 ± 0.50	79.56 ± 0.34	78.75 ± 0.27	79.15 ± 0.34	<u>81.20 ± 0.41</u>	78.98 ± 0.45	**81.27 ± 0.31**
VGG-16	Cifar-10	0.1	62.28 ± 5.72	63.19 ± 5.15	65.97 ± 3.82	62.00 ± 3.19	64.78 ± 5.69	65.52 ± 4.96	<u>69.15 ± 2.16</u>	**74.49 ± 0.92**
		0.5	78.82 ± 0.21	78.49 ± 0.26	78.98 ± 0.11	78.09 ± 0.48	79.30 ± 0.43	79.12 ± 0.25	<u>80.07 ± 0.19</u>	**84.24 ± 0.37**
		1.0	79.53 ± 0.34	79.52 ± 0.30	80.08 ± 0.24	79.67 ± 0.45	79.76 ± 0.20	80.08 ± 0.32	<u>80.85 ± 0.99</u>	**85.12 ± 0.13**
		IID	79.96 ± 0.05	79.79 ± 0.07	80.13 ± 0.05	79.66 ± 0.06	79.89 ± 0.05	80.66 ± 0.09	<u>82.20 ± 0.05</u>	**85.66 ± 0.15**
	Cifar-100	0.1	47.29 ± 0.96	48.02 ± 0.68	49.11 ± 1.60	47.38 ± 1.47	49.36 ± 0.55	48.78 ± 1.03	<u>51.30 ± 1.00</u>	**55.33 ± 0.72**
		0.5	55.60 ± 0.55	55.45 ± 0.70	56.29 ± 0.84	54.45 ± 0.58	56.40 ± 0.47	56.73 ± 0.41	<u>58.02 ± 0.31</u>	**65.07 ± 0.25**
		1.0	56.05 ± 0.45	55.75 ± 0.36	57.96 ± 0.32	55.70 ± 0.37	56.69 ± 0.25	57.46 ± 0.25	<u>58.53 ± 0.31</u>	**65.66 ± 0.15**
		IID	57.22 ± 0.28	56.65 ± 0.23	58.47 ± 0.16	57.33 ± 0.17	57.55 ± 0.16	57.96 ± 0.11	<u>58.62 ± 0.08</u>	**66.33 ± 0.10**
	FEMNIST	−	83.96 ± 0.43	84.27 ± 0.32	84.39 ± 0.28	83.64 ± 0.27	83.99 ± 0.32	<u>84.59 ± 0.21</u>	83.97 ± 0.57	**85.36 ± 0.21**

(a) $\alpha = 0.1$ (b) $\alpha = 0.5$ (c) $\alpha = 1.0$ (d) IID
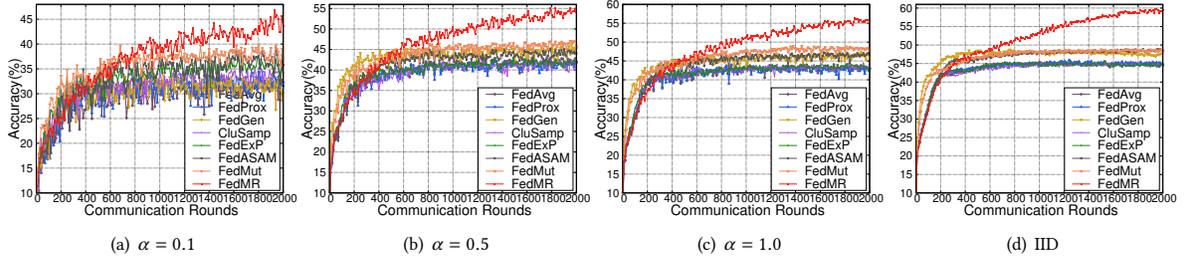
Figure 9: Learning curves of FL methods based on ResNet-20 model for CIFAR-100 dataset.

on clients are non-IID distributed ($\alpha = 1.0$). Note that all the cases here did not use the two-stage training scheme. From this figure, we can find that FedMR outperforms the other variants significantly. Moreover, when the granularity of partitioning goes coarser, the classification performance of FedMR becomes worse.

5.4.3 Two-stage Training Scheme. To demonstrate the effectiveness of our proposed two-stage training scheme, we conducted experiments on CIFAR-10 dataset using ResNet-20-based and VGG-16-based FedMR, where the data on clients are non-IID distributed ($\alpha = 1.0$). Figure 12 presents the learning trends of FedMR with

five different two-stage training settings. Here, we use the notation "FedMR-*n*" to denote that the first stage involves *n* rounds of model aggregation-based local training to obtain a pre-trained global model, while the remaining rounds conduct local training based on our proposed model recombination-based method. From Figure 12, we can observe that the two-stage training-based FedMR methods (i.e., FedMR-50 and FedMR-100) achieve the best performance from the perspectives of test accuracy and convergence rate. Note that our two-stage training scheme can achieve a more significant improvement on the case using VGG-16 model, which has a much larger size than ResNet-20 model. This is because the

Is Aggregation the Only Choice? Federated Learning via Layer-wise Model Recombination

KDD '24, August 25–29, 2024, Barcelona, Spain.

(a) $K = 5$ (b) $K = 10$ (c) $K = 20$ (d) $K = 50$ (e) $K = 100$
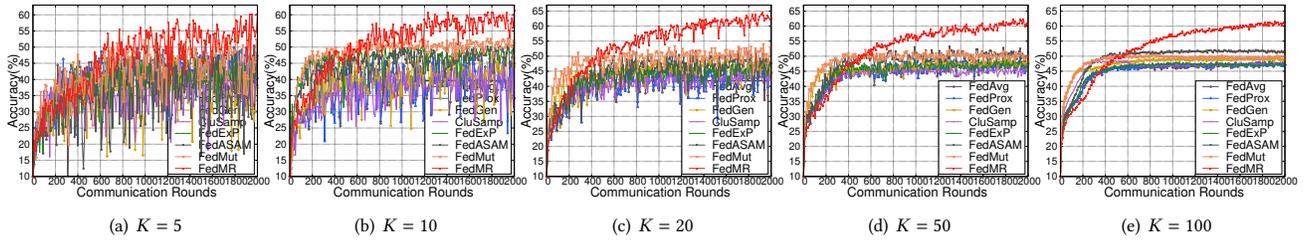
Figure 10: Comparison of FL methods using ResNet-20 model on CIFAR-10 dataset with $\alpha = 0.1$.

(a) ResNet-20   (b) VGG-16
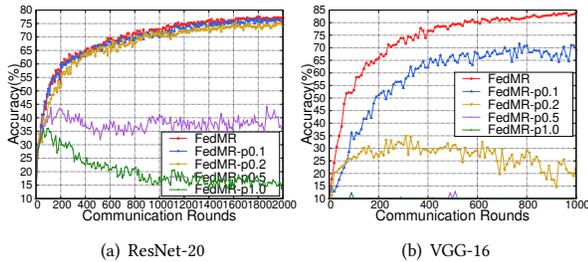
Figure 11: Learning curves for partitioning strategies.

(a) ResNet-20   (b) VGG-16
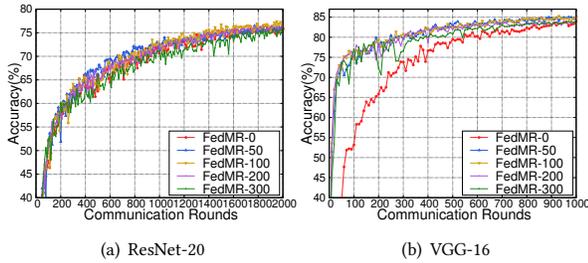
Figure 12: Learning curves for two-stage training settings.

fine-grained FedMR without the first-stage training is not good at dealing with large-size models at the beginning of FL training, which requires many more training rounds than the coarse-grained aggregation-based methods to achieve a given preliminary classification accuracy target. By resorting to the two-stage training scheme, such a slow convergence problem can be greatly mitigated.

5.5 Discussions

5.5.1 Privacy Preserving. Similar to traditional FedAvg-based FL methods, FedMR does not require clients to send their data to the cloud server, thus the data privacy can be mostly guaranteed by the secure clients themselves. Since our model recombination operation breaks the dependencies between model layers and conducts the shuffling of layers among models, in practice, it is hard for adversaries to restore the confidential data from a fragmentary recombined model without knowing the sources of layers. We present a secure recombination mechanism to avoid privacy leakage from the cloud server. Please see Appendix B for more details. Our secure recombination mechanism ensures that the cloud server only

receives recombined models from clients, which means that the cloud server cannot restore the model of each client.

5.5.2 Limitations. As a novel FL paradigm, FedMR shows much better inference performance than most SOTA FL methods. Although this paper proposed an efficient two-stage training scheme to accelerate the overall FL training processes, there still exist numerous chances (e.g., client selection strategies, dynamic combination of model aggregation and model recombination operations) to enable further optimization on the current version of FedMR. Meanwhile, the current version of FedMR does not consider personalization [31, 33] and fairness [26, 27], two very important topics that deserve to be studied in the future.

6 Conclusion

Due to the coarse-grained aggregation of FedAvg as well as the uniform client model initialization, when dealing with uneven data distribution among clients, existing Federated Learning (FL) methods greatly suffer from the problem of low inference performance. To address this problem, this paper presented a new FL paradigm named FedMR, which enables different layers of local models to be trained on different clients. Since FedMR supports both fine-grained model recombination and diversified local training initialization, it enables effective and efficient search for superior generalized models for all clients. Comprehensive experimental results show both the effectiveness and pervasiveness of our proposed method in terms of inference accuracy and convergence rate.

7 Acknowledgement

References

[1] Durmus Alp Emre Acar, Yue Zhao, Ramon Matas, Matthew Mattina, Paul Whatmough, and Venkatesh Saligrama. 2020. Federated Learning Based on Dynamic Regularization. In *Proc. of International Conference on Learning Representations*.

[2] Mohammed Adnan, Shivam Kalra, Jesse C Cresswell, Graham W Taylor, and Hamid R Tizhoosh. 2022. Federated learning and differential privacy for medical image analysis. *Scientific reports* 12, 1 (2022), 1953.

[3] Naman Agarwal, Peter Kairouz, and Ziyu Liu. 2021. The skellam mechanism for differentially private federated learning. *Advances in Neural Information Processing Systems* 34 (2021), 5052–5064.

[4] Samiul Alam, Luyang Liu, Ming Yan, and Mi Zhang. 2022. Fedrolex: Model-heterogeneous federated learning with rolling sub-model extraction. *Advances in neural information processing systems* 35 (2022), 29677–29690.

[5] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. 2017. Practical secure aggregation for privacy-preserving machine learning. In *Proc. of ACM SIGSAC Conference on Computer and Communications Security*. 1175–1191.

[6] Debora Caldarola, Barbara Caputo, and Marco Ciccone. 2022. Improving generalization in federated learning by seeking flat minima. In *Proc. of European Conference on Computer Vision*. 654–672.

[7] Sebastian Caldas, Sai Meher Karthik Duddu, Peter Wu, Tian Li, Jakub Konečnỳ, H Brendan McMahan, Virginia Smith, and Ameet Talwalkar. 2018. Leaf: A benchmark for federated settings. *arXiv preprint arXiv:1812.01097* (2018).

[8] Cheng Chen, Ziyi Chen, Yi Zhou, and Bhavya Kailkhura. 2020. Fedcluster: Boosting the convergence of federated learning via cluster-cycling. In *Proc. of IEEE International Conference on Big Data*. 5017–5026.

[9] Yann Fraboni, Richard Vidal, Laetitia Kameni, and Marco Lorenzi. 2021. Clustered sampling: Low-variance and improved representativity for clients selection in federated learning. In *Proc. of International Conference on Machine Learning*. 3407–3416.

[10] Moritz Hardt, Ben Recht, and Yoram Singer. 2016. Train faster, generalize better: Stability of stochastic gradient descent. In *Proc. of International Conference on Machine Learning*. 1225–1234.

[11] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Flat minima. *Neural computation* 9, 1 (1997), 1–42.

[12] Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown. 2019. Measuring the effects of non-identical data distribution for federated visual classification. *arXiv preprint arXiv:1909.06335* (2019).

[13] Ming Hu, E Cao, Hongbing Huang, Min Zhang, Xiaohong Chen, and Mingsong Chen. 2023. AIoTML: A Unified Modeling Language for AIoT-Based Cyber-Physical Systems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (2023).

[14] Ming Hu, Yue Cao, Anran Li, Zhiming Li, Chengwei Liu, Tianlin Li, Mingsong Chen, and Yang Liu. 2024. FedMut: Generalized Federated Learning via Stochastic Mutation. In *Proc. of the AAAI Conference on Artificial Intelligence*, Vol. 38. 12528–12537.

[15] Ming Hu, Wenxue Duan, Min Zhang, Tongquan Wei, and Mingsong Chen. 2020. Quantitative timing analysis for cyber-physical systems using uncertainty-aware scenario-based specifications. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 39, 11 (2020), 4006–4017.

[16] Ming Hu, Zeke Xia, Dengke Yan, Zhihao Yue, Jun Xia, Yihao Huang, Yang Liu, and Mingsong Chen. 2023. GitFL: Uncertainty-Aware Real-Time Asynchronous Federated Learning Using Version Control. In *Proc. of IEEE Real-Time Systems Symposium*. 145–157.

[17] Yutao Huang, Lingyang Chu, Zirui Zhou, Lanjun Wang, Jiangchuan Liu, Jian Pei, and Yong Zhang. 2021. Personalized Cross-Silo Federated Learning on Non-IID Data. In *Proc. of the AAAI Conference on Artificial Intelligence*.

[18] Fatih Ilhan, Gong Su, and Ling Liu. 2023. Scalefl: Resource-adaptive federated learning with heterogeneous clients. In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 24532–24541.

[19] Divyansh Jhunjhunwala, Shiqiang Wang, and Gauri Joshi. 2023. Fedexp: Speeding up federated averaging via extrapolation. *arXiv preprint arXiv:2301.09604* (2023).

[20] Chentao Jia, Ming Hu, Zekai Chen, Yanxin Yang, Xiaofei Xie, Yang Liu, and Mingsong Chen. 2024. AdaptiveFL: Adaptive Heterogeneous Federated Learning for Resource-Constrained AIoT Systems. In *Proc. of Design Automation Conference*. 1–6.

[21] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. 2021. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning* 14, 1–2 (2021), 1–210.

[22] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. 2020. SCAFFOLD: Stochastic controlled averaging for federated learning. In *Proc. of International Conference on Machine Learning*. 5132–5143.

[23] Alex Krizhevsky. 2009. Learning Multiple Layers of Features from Tiny Images. *Master's thesis, University of Tront* (2009).

[24] Jungmin Kwon, Jeongseop Kim, Hyunseo Park, and In Kwon Choi. 2021. Asam: Adaptive sharpness-aware minimization for scale-invariant learning of deep neural networks. In *Proc. of International Conference on Machine Learning*. PMLR, 5905–5914.

[25] Anran Li, Rui Liu, Ming Hu, Luu Anh Tuan, and Han Yu. 2023. Towards interpretable federated learning. *arXiv preprint arXiv:2302.13473* (2023).

[26] Tianlin Li, Qing Guo, Aishan Liu, Mengnan Du, Zhiming Li, and Yang Liu. 2023. FAIRER: fairness as decision rationale alignment. In *Proc. of International Conference on Machine Learning*. 19471–19489.

[27] Tianlin Li, Zhiming Li, Anran Li, Mengnan Du, Aishan Liu, Qing Guo, Guozhu Meng, and Yang Liu. 2023. Fairness via group contribution matching. In *Proc. of International Joint Conference on Artificial Intelligence*. 436–445.

[28] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. 2020. Federated optimization in heterogeneous networks. *Proc. of Machine Learning and Systems* 2 (2020), 429–450.

[29] Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. 2020. On the Convergence of FedAvg on Non-IID Data. In *Proc. of International Conference on Learning Representations*.

[30] Tao Lin, Lingjing Kong, Sebastian U Stich, and Martin Jaggi. 2020. Ensemble distillation for robust model fusion in federated learning. *Advances in Neural Information Processing Systems* 33 (2020), 2351–2363.

[31] Xiaosong Ma, Jie Zhang, Song Guo, and Wenchao Xu. 2022. Layer-wised model aggregation for personalized federated learning. In *Proc. of the IEEE/CVF conference on computer vision and pattern recognition*. 10092–10101.

[32] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *Proc. of Artificial intelligence and statistics*. 1273–1282.

[33] Krishna Pillutla, Kshitiz Malik, Abdel-Rahman Mohamed, Mike Rabbat, Maziar Sanjabi, and Lin Xiao. 2022. Federated learning with partial model personalization. In *Proc. of International Conference on Machine Learning*. 17716–17758.

[34] Sebastian Urban Stich. 2019. Local SGD Converges Fast and Communicates Little. In *Proc. of International Conference on Learning Representations*.

[35] Ben Tan, Bo Liu, Vincent Zheng, and Qiang Yang. 2020. A federated recommender system for online services. In *Proc. of ACM Conference on Recommender Systems*. 579–581.

[36] TorchvisionModel. 2019. Models and pre-trained weight. https://pytorch.org/vision/stable/models.html.

[37] Haozhao Wang, Yabo Jia, Meng Zhang, Qinghao Hu, Hao Ren, Peng Sun, Yonggang Wen, and Tianwei Zhang. 2024. FedDSE: Distribution-aware Sub-model Extraction for Federated Learning over Resource-constrained Devices. In *Proc. of the ACM on Web Conference*. 2902–2913.

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

[39] Haozhao Wang, Yichen Li, Wenchao Xu, Ruixuan Li, Yufeng Zhan, and Zhigang Zeng. 2023. Dafkd: Domain-aware federated knowledge distillation. In *Proc. of the IEEE/CVF conference on Computer Vision and Pattern Recognition*. 20412–20421.

[40] Haozhao Wang, Haoran Xu, Yichen Li, Yuan Xu, Ruixuan Li, and Tianwei Zhang. 2023. FedCDA: Federated Learning with Cross-rounds Divergence-aware Aggregation. In *Proc. of International Conference on Learning Representations*.

[41] Jiaqi Wang, Suhan Cui, and Fenglong Ma. 2023. FedLEGO: Enabling Heterogenous Model Cooperation via Brick Reassembly in Federated Learning. In *Proc. of International Workshop on Federated Learning for Distributed Data Mining*.

[42] Jie Wen, Zhixia Zhang, Yang Lan, Zhihua Cui, Jianghui Cai, and Wensheng Zhang. 2023. A survey on federated learning: challenges and applications. *International Journal of Machine Learning and Cybernetics* 14, 2 (2023), 513–535.

[43] Dongxian Wu, Shu-Tao Xia, and Yisen Wang. 2020. Adversarial weight perturbation helps robust generalization. *Advances in Neural Information Processing Systems* 33 (2020), 2958–2969.

[44] Zeke Xia, Ming Hu, Dengke Yan, Xiaofei Xie, Tianlin Li, Anran Li, Junlong Zhou, and Mingsong Chen. 2024. CaBaFL: Asynchronous Federated Learning via Hierarchical Cache and Feature Balance. *arXiv preprint arXiv:2404.12850* (2024).

[45] 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).

[46] Qian Yang, Jianyi Zhang, Weituo Hao, Gregory P Spell, and Lawrence Carin. 2021. Flop: Federated learning on medical datasets using partial networks. In *Proc. of the ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 3845–3853.

[47] Xinqian Zhang, Ming Hu, Jun Xia, Tongquan Wei, Mingsong Chen, and Shiyan Hu. 2020. Efficient federated learning for cloud-based AIoT applications. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 40, 11 (2020), 2211–2223.

[48] Zhuangdi Zhu, Junyuan Hong, and Jiayu Zhou. 2021. Data-free knowledge distillation for heterogeneous federated learning. In *Proc. of International Conference on Machine Learning*. 12878–12889.

Is Aggregation the Only Choice? Federated Learning via Layer-wise Model Recombination

KDD '24, August 25–29, 2024, Barcelona, Spain.

A Proof of FedMR Convergence

A.1 Notations

In our FedMR approach, the global model is aggregated from all the recombined models and all the models have the same weight. Let t exhibit the tth SGD iteration on the local device, v is the intermediate variable that represents the result of SGD update after exactly one iteration. The update of FedMR is as follows:

$$v_{t+1}^k = w_t^k - \eta_t \nabla f_k(w_t^k, \xi_t^k), \tag{1}$$

$$w_{t+1}^k = \begin{cases} v_{t+1}^k, & if \quad E \nmid t+1 \\ RM(v_{t+1}^k), & if \quad E \mid t+1 \end{cases}, \tag{2}$$

where w_t^k represents the model of the kth client in the tth iteration. w_{t+1} denotes the global model of the $(t+1)^{th}$ iteration. $RM(v_{t+1}^k)$ denotes the recombined model.

Since FedMR recombines all the local models in each round and the recombination only shuffles layers of models, the parameters of recombined models are all from the models before recombination, and no parameters are discarded. Therefore, when $E \mid t+1$, we can obtain the following invariants:

$$\sum_{k=1}^K v_{t+1}^k = \sum_{k=1}^K RM(v_{t+1}^k) = \sum_{k=1}^K w_{t+1}^k, \tag{3}$$

$$\sum_{k=1}^K ||v_{t+1}^k - x||^2 = \sum_{k=1}^K ||w_{t+1}^k - x||^2, \tag{4}$$

where w_t^k is the kth recombined model in $(t-1)^{th}$ iteration, which is as the local model to be dispatched to k^{th} client in t^{th} iteration, x can any vector with the same size as v_t^k. Similar to [34], we define two variables $\bar{v}_t$ and $\bar{w}_t$:

$$\bar{v}_t = \frac{1}{K}\sum_{k=1}^K v_t^k, \bar{w}_t = \frac{1}{K}\sum_{k=1}^k w_t^k. \tag{5}$$

Inspired by [29], we make the following definition:

$$g_t^k = \nabla f_k(w_t^k; \xi_t^k). \tag{6}$$

A.2 Proof of Lemma 4.4

PROOF. Assume v_t^k has n layers, we have $v_t^k = L_1 \oplus L_2 \oplus ... \oplus L_n$. Let $L_i = [p_{(i,0)}^{v_t^k}, p_{(i,1)}^{v_t^k}, ..., p_{(i,|L_i|)}^{v_t^k}]$, where $p_{(i,j)}^{v_t^k}$ denotes the jth parameter of the layer L_i in the model v_t^k. We have

$$\sum_{k=1}^K ||v_t^k - x||^2 = \sum_{k=1}^K \sum_{i=1}^n \sum_{j=1}^{|L_i|} ||p_{(i,j)}^{v_t^k} - p_{(i,j)}^x||^2 \tag{7}$$

$$\sum_{k=1}^K ||w_t^k - x||^2 = \sum_{k=1}^K \sum_{i=1}^n \sum_{j=1}^{|L_i|} ||p_{(i,j)}^{w_t^k} - p_{(i,j)}^x||^2 \tag{8}$$

Since model recombination only shuffles layers of models, the parameters of recombined models are all from the models before recombination and no parameters are discarded. We have

$$\forall_{i \in [1,n], j \in [1,|L_i|]} \sum_{k=1}^K p_{(i,j)}^{v_t^k} = \sum_{k=1}^K p_{(i,j)}^{w_t^k} \tag{9}$$

$$\forall_{k \in [1,K], i \in [1,n], j \in [1,|L_i|]} \exists_{q \in [1,K]} \{ p_{(i,j)}^{v_t^k} = p_{(i,j)}^{w_t^q} \} \tag{10}$$

$$\forall_{k \in [1,K], i \in [1,n], j \in [1,|L_i|]} \exists_{q \in [1,K]} \{ p_{(i,j)}^{w_t^k} = p_{(i,j)}^{v_t^q} \} \tag{11}$$

According to Equations 9-11, we have

$$\sum_{k=1}^K ||v_t^k - x||^2 = \sum_{k=1}^K \sum_{i=1}^n \sum_{j=1}^{|L_i|} ||p_{(i,j)}^{v_t^k} - p_{(i,j)}^x||^2$$
$$= \sum_{k=1}^K ||w_t^k - x||^2 \tag{12}$$

$\square$

A.3 Key Lemmas

To facilitate the proof of our Theorem 1, inspired by [29], we can present the following two lemmas. Note that the following proofs are general proofs for all the multi-model-based FL approaches that satisfy Lemma 4.4.

Lemma A.1. (Results of one step SGD). If $\eta_t \leq \frac{1}{4L}$, we have

$$\mathbb{E}||\bar{v}_{t+1} - w^\star||^2 \leq \frac{1}{K}\sum_{k=1}^K (1-\mu\eta_t)||v_t^k - w^\star||^2$$
$$+ \frac{1}{K}\sum_{k=1}^K ||w_t^k - w_{t_0}^k||^2 + 10\eta_t^2 L\Gamma$$

PROOF. According to Lemma 4.4 (i.e., Equation 3 and Equation 4), we have

$$||\bar{v}_{t+1} - w^\star||^2 \leq \frac{1}{K}\sum_{k=1}^K ||v_{t+1}^k - w^\star||^2$$
$$= \frac{1}{K}\sum_{k=1}^K (||v_t^k - w^\star||^2 - 2\eta_t\langle w_t^k - w^\star, g_t^k\rangle \tag{13}$$
$$+ \eta_t^2 ||g_t^k||^2)$$

Let $B_1 = -2\eta_t\langle w_t^k - w^\star, g_t^k\rangle$ and $B_2 = \eta_t^2 \sum_{k=1}^K ||g_t^k||^2$. According to Assumption 4.2, we have

$$B_1 \leq -2\eta_t(f_k(w_t^k) - f_k(w^\star)) - \mu\eta_t||w_t^k - w^\star||^2 \tag{14}$$

According to Assumption 4.1, we have

$$B_2 \leq 2\eta_t^2 L(f_k(w_t^k) - f_k^\star) \tag{15}$$

According to Equation 14 and 15, we have

$$||\bar{v}_{t+1} - w^\star||^2 \leq \frac{1}{K}\sum_{k=1}^K [(1-\mu\eta_t)||v_t^k - w^\star||^2$$
$$- 2\eta_t(f_k(w_t^k) - f_k(w^\star)) + 2\eta_t^2 L(f_k(w_t^k) - f_k^\star)] \tag{16}$$

Let $C = \frac{1}{K}\sum_{k=1}^K [-2\eta_t(f_k(w_t^k) - f_k(w^\star)) + 2\eta_t^2 L(f_k(w_t^k) - f_k^\star)]$. We have

$$C = \frac{-2\eta_t}{K}\sum_{k=1}^K (f_k(w_t^k) - f_k(w^\star)) + \frac{2\eta_t^2 L}{K}\sum_{k=1}^K (f_k(w_t^k) - f_k^\star)$$
$$= -\frac{2\eta_t(1-\eta_t L)}{K}\sum_{k=1}^K (f_k(w_t^k) - f^\star) + \frac{2\eta_t^2 L}{K}\sum_{k=1}^K (f^\star - f_k^\star) \tag{17}$$

Let $\Gamma = f^\star - \frac{1}{K}\sum_{k=1}^{K} f_k^\star$ and $\phi = 2\eta_t(1 - L\eta_t)$. We have

$$C = -\frac{\phi}{K}\sum_{k=1}^{K}(f_k(w_t^k) - f^\star) + 2\eta_t^2 L\Gamma \qquad (18)$$

Let $D = -\frac{1}{K}\sum_{k=1}^{K}(f_k(w_t^k) - f^\star)$, $E \mid t_0$ and $t - t_0 \leq E$. We have

$$D = -\frac{1}{K}\sum_{k=1}^{K}(f_k(w_t^k) - f_k(w_{t_0}^k) + f_k(w_{t_0}^k) - f^\star) \qquad (19)$$

By Cauchy–Schwarz inequality, we have

$$
\begin{aligned}
D &\leq \frac{1}{2K}\sum_{k=1}^{K}(\eta_t\|\nabla f_k(w_{t_0}^k)\|^2 + \frac{1}{\eta_t}\|w_t^k - w_{t_0}^k\|^2) \\
&\quad - \frac{1}{K}\sum_{k=1}^{K}(f_k(w_{t_0}^k) - f^\star) \\
&\leq \frac{1}{2K}\sum_{k=1}^{K}[2\eta_t L(f_k(w_{t_0}^k) - f_k^\star) + \frac{1}{\eta_t}\|w_t^k - w_{t_0}^k\|^2] \\
&\quad - \frac{1}{K}\sum_{k=1}^{K}(f_k(w_{t_0}^k) - f^\star)
\end{aligned}
\qquad (20)
$$

Note that since $\eta \leq \frac{1}{4L}$, $\eta_t \leq \phi \leq 2\eta_t$ and $\eta_t L \leq \frac{1}{4}$. According to Equation 20, we have

$$
\begin{aligned}
C &\leq \frac{\phi}{2\eta_t K}\sum_{k=1}^{K}\|w_t^k - w_{t_0}^k\|^2 + (\phi\eta_t L + 2\eta_t^2 L)\Gamma + \frac{\phi}{K}\sum_{k=1}^{K}(f^\star - f_k^\star) \\
&\leq \frac{\phi}{2\eta_t K}\sum_{k=1}^{K}\|w_t^k - w_{t_0}^k\|^2 + (\phi\eta_t L + \phi + 2\eta_t^2 L)\Gamma \\
&\leq \frac{1}{K}\sum_{k=1}^{K}\|w_t^k - w_{t_0}^k\|^2 + 10\eta_t^2 L\Gamma
\end{aligned}
\qquad (21)
$$

$\square$

Lemma A.2. *According to Equation 2, the model recombination occurs every E iterations. Assume that in each training round, t_0 is the first iteration and iteration $t - t_0 \leq E - 1$. Given the constraint on learning rate from [29], we know that $\eta_t \leq \eta_{t_0} \leq 2\eta_t$. It follows that*

$$\frac{1}{K}\sum_{k=1}^{K}\|w_t^k - w_{t_0}^k\|^2 \leq 4\eta_t^2(E-1)^2 G^2.$$

PROOF.

$$
\begin{aligned}
\frac{1}{K}\sum_{k=1}^{K}\|w_t^k - w_{t_0}^k\|^2 &= \frac{1}{K}\sum_{k=1}^{K}\|\sum_{t=t_0}^{t_0+E-1}\eta_t \nabla f_{a_1}(w_t^{a_1}; \xi_t^{a_1})\|^2 \\
&\leq (t - t_0)\sum_{t=t_0}^{t_0+E-1}\eta_t^2 G^2 \\
&\leq (E - 1)\sum_{t=t_0}^{t_0+E-1}\eta_t^2 G^2 \\
&\leq 4\eta_t^2(E-1)^2 G^2.
\end{aligned}
$$

$\square$

A.4 Proof of Theorem 1

Based on Lemmas A.1 and A.2, we can prove Theorem 1 using the proof framework of FedAvg [29]. Due to space limitations, please refer to the proof of FedAvg [29] for the details.

B Secure Model Recombination Mechanism

To avoid the risk of privacy leakage caused by exposing gradients or models to the cloud server, we propose a secure model recombination mechanism for FedMR, which allows the random exchange of model layers among clients before model training or upload. As shown in Figure 13, within a round of the secure model recombination, the update of each model (i.e., m) consists of four stages:
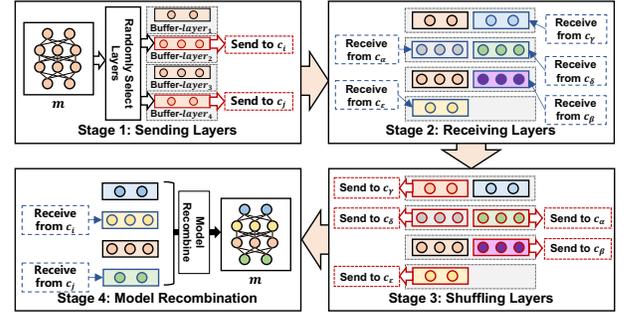
Figure 13: Workflow of secure model recombination.

Stage 1: Assume that the local model has *len* layer. Each client maintains a buffer for each layer. Firstly, each client randomly selects a part of its layers and sends them to other activated clients, while the remaining layers are saved in their corresponding buffers. Note that a selected layer can only be sent to one client. For example, in Figure 13, the client m sends $layer_2$ and $layer_4$ to c_i and c_j, respectively.

Stage 2: Once receiving a layer from another client, the receiving client m will add the layer to its corresponding buffer. For example, in Figure 13, the client m totally receives five layers. Besides the retained two layers in stage 1, m now has seven layers in total in its buffers.

Stage 3: For each layer buffer of m, if there contains one element received from a client c in stage 2, our mechanism will randomly select one layer in the buffer and return it back to c. For example, in Figure 13, m randomly returns a layer in *Buffer-layer1* back to a client c_γ.

Stage 4: Once receiving the returned layers from other clients, our mechanism will recombine them with all the other layers in the buffers to form a new model. Note that the recombined model may significantly differ from the original model in Stage 1.

Note that each FL training round can perform multiple times secure model recombination. Due to the randomness, it is hard for adversaries to figure out the sources of client model layers. In addition, the cloud server will broadcast a public key before the secure recombination to prevent privacy leakage. By using the public key to encrypt the model parameters of each layer, the other clients cannot directly obtain their received parameters.