Device-Performance-Driven Heterogeneous Multiparty Learning for Arbitrary Images

Yuanqiao Zhang, Maoguo Gong, *Senior Member, IEEE*, Yuan Gao, A. K. Qin, *Senior Member, IEEE*, Kun Wang, Yiming Lin, and Shanfeng Wang, *Member, IEEE*

***Abstract*— Multiparty learning (MPL) is an emerging framework for privacy-preserving collaborative learning. It enables individual devices to build a knowledge-shared model and remaining sensitive data locally. However, with the continuous increase of users, the heterogeneity gap between data and equipment becomes wider, which leads to the problem of model heterogeneous. In this article, we concentrate on two practical issues: data heterogeneous problem and model heterogeneous problem, and propose a novel personal MPL method named device-performance-driven heterogeneous MPL (HMPL). First, facing the data heterogeneous problem, we focus on the problem of various devices holding arbitrary data sizes. We introduce a heterogeneous feature-map integration method to adaptively unify the various feature maps. Meanwhile, to handle the model heterogeneous problem, as it is essential to customize models for adapting to the various computing performances, we propose a layer-wise model generation and aggregation strategy. The method can generate customized models based on the device's performance. In the aggregation process, the shared model parameters are updated through the rules that the network layers with the same semantics are aggregated with each other. Extensive experiments are conducted on four popular datasets, and the result demonstrates that our proposed framework outperforms the state of the art (SOTA).**

***Index Terms*— Data and model heterogeneity, multiparty learning (MPL), personalized model generation, pyramidal aggregation strategy.**

NOMENCLATURE

$f(x)(\boldsymbol{F}(\boldsymbol{x}))$	Set of objective function.
W, w	Set of model parameters.
C, c	Number (index) of clients.
t	Communication round.
D_c	Local data of the client c.
$\mathcal{P}_c$	Data point index of client c.
K_B	Total number of basic layer (BS Layer).
K_I	Total number of isolated personal layer (IP Layer).
η	Learning rate.

Manuscript received 2 September 2022; revised 13 February 2023; accepted 28 May 2023. This work was supported in part by the Natural Science Foundation of China under Grant 62036006 and Grant 62276200 and in part by the Key-Area Research and Development Program of Guangdong Province under Grant 2020B090921001. *(Corresponding author: Maoguo Gong.)*

Yuanqiao Zhang, Maoguo Gong, Yuan Gao, Kun Wang, Yiming Lin, and Shanfeng Wang are with the Key Laboratory of Collaborative Intelligence Systems, Ministry of Education, Xidian University, Xi'an, Shaanxi 710071, China (e-mail: zhangyq@stu.xidian.edu.cn; gong@ieee.org; cn_gaoyuan@foxmail.com; m15009208535@163.com; 1169086366@qq.com; sfwang@xidian.edu.cn).

A. K. Qin is with the Department of Computing Technologies, Swinburne University of Technology, Melbourne, VIC 3122, Australia (e-mail: kqin@swin.edu.au).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2023.3282242.

Digital Object Identifier 10.1109/TNNLS.2023.3282242

I. INTRODUCTION

IN RECENT years, as more and more data are generated on decentralized devices [1], the mobile phone has gradually become the most popular data carriers [2]. However, big-data processing methods always require centralized data, and this situation leads to the data sharing problem [3]. Because of the data privacy and security policies, such as the General Data Protection Regulation (GDPR) [4] and Health Insurance Portability and Accountability Act (HIPAA) [5], it is inconvenient to achieve complete data sharing. Under this circumstance, there also exists another crucial issue: if the interactive process is broadcasted or up-downloaded, the communication cost will dramatically increase. Therefore, it is essential to invent a fashion for handling such specific circumstances.

A method provided by the major service providers called multiparty learning (MPL) [6] has been deployed in recent years. One of the most distinctive advantages is that, it can smoothly engage privacy-sensitive problems [7] and distributed edge-data problems. By establishing a multitrusted central server, the devices can communicate and coordinate with each other. The models of these clients are trained by their local data, and all the updates are uploaded and aggregated on server side for collaborative optimization. It is worth noting that, the whole process does not require sensitive data to leave the client, and this builds a privacy-preserving application.

MPL has been widely used in industry [8], but one limitation is that, it always assumes one single central model to efficiently handle all the devices [9]. It may be difficult to access in reality, since different data owners may hold various data distributions as is shown in Fig. 1 as an example. To handle such kinds of problems, a novel technique called personalized MPL has been proposed [10]. Three challenges faced by personalized MPL systems are listed as follows [11].

1) *Data Heterogeneity [12]:* For example, people speaking different languages will lead to various text data, and pictures collected by different devices may be at

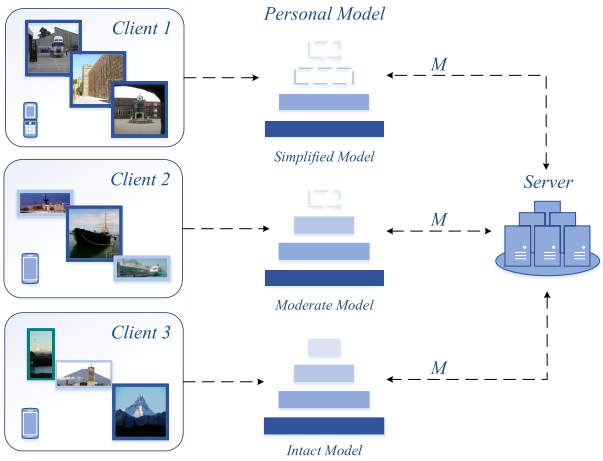


Fig. 1. Various types of devices hold various sizes of data. The different colors of the photo frames saved in the different clients (displayed on the left) represent different photo sizes, such as 1:1, 4:3, and so on. In addition, different devices will adopt models with different structures (displayed in the middle), such as high-performance devices deploying models with more complex structures. It is reasonable to assign more appropriate models to systems of heterogeneous equipment.

different locations or with different sizes. Hence, the data may be not identically distributed or unbalanced (non-IID).

2) *Model Heterogeneity [10]:* Separate clients need specific models customized for the personal environment. Another common situation is that, old equipment may have difficulties in dealing with new complex models.
3) *Device Heterogeneity [11]:* Various devices have differences in terms of storage, computation, and communication capabilities [13].

Considering these challenges, the problems are confronted with new complexities and are worthy to study.

In daily life, mobile phone devices can be regarded as potential application examples of MPL. A large number of users generate large amounts of data. For instance, one common situation in daily life is that, there always exists arbitrary sizes of images. For example, the download images are always with a 1:1 aspect ratio, while the photos taken by the camera are 4:3, sometimes 16:9. This circumstance leads to a technical issue of convolutional neural networks (CNNs): it is unworkable to utilize the unitary model for training different sizes of the data [14]. In fact, the convolutional layers can use arbitrary data to generate feature maps of any size, but the fully connected (FC) layers require the input of fixed size according to their definition. Meanwhile, even if the model can be trained on a single device, it will cause a great issue when aggregating on the global server. Traditional methods always fit the input size via cropping [15] or warping [16], [17]. But, these methods may not contain the entire object or may result in unwanted geometric distortion.

At the same time, the diversity of data and devices will lead to the model heterogeneity problem. For example, when analyzing arbitrary-size data and using the same form of the model, an overcomplex model may lead to excessive pressure on weak devices. This problem has practical significance but it will become more complex at the same time. Chen et al. [18] designed an asynchronous model update method to tackle the weak device problem, but this method only considered the situation of an isomorphic model. Meanwhile, focusing on high-performance clients, we also want to maximize the calculation power. Inspired by this, it is rational to propose a more sensitive personalized MPL strategy for customized model generation and aggregation.

In this work, we propose a novel MPL framework focusing on the arbitrary image size problem and customized model problem. For the subject of different sizes, we introduce the concept of spatial pyramid pooling (SPP) [14] to homogenize the contradictions caused by different dimension sizes. To achieve the ambition of maximizing all the computing utilization while taking care of the weak devices, we propose a novel *pyramidal* strategy inspired by the feature pyramid network (FPN) [19] for efficient generation and aggregation of each customized model.

In summary, our contribution can be concluded as follows.

1) In the data heterogeneity scenarios, we introduce a heterogeneous feature-map integration method. This method can eliminate the calculation difficulties caused by different sizes while facilitating model aggregation on the server side.
2) To solve the model heterogeneity problem, we propose a novel pyramidal MPL strategy. In this strategy, the local models are generated personally and aggregated by similar semantic characteristics.
3) We analyze our method with existing state-of-the-art (SOTA) personalized MPL algorithms, and the experimental results show the improvement of the performances on the popular datasets.

The rest of this article is constructed as follows. Section II introduces the relevant background. In Section III, the main framework of the algorithm is proposed in detail. The experimental results and the ablation discussion are summarized in Section IV. Finally, the main conclusion is described in Section V.

II. Preliminaries

In this section, the correlation methods of MPL are briefly expounded. Two popular techniques related to our framework, including SPP networks and FPNs, are specifically discussed.

A. Multiparty Learning

MPL [8] is a general conception of distributed machine learning. Different from both edge computing [20], which allows the transfer of raw data, and federated learning [21], which emphasizes privacy in the transmission process, MPL places a greater emphasis on collaboration between users to optimize a global model. It facilitates multiple devices collaboratively training a sharable model while keeping the sensitive data locally [6]. A central server is settled to coordinate the learning process consisting of multiple rounds. At the beginning of each communication round, the global server will send the current global model to participating devices. Each device trains the model of its private data and

uploads to the global server. The processing of a general MPL framework can be described mathematically as follows:

$$\min_{w} \; l(w) = \sum_{c=1}^{C} \frac{p_c}{p} L_c(w)$$
$$L_c(w) = \frac{1}{p_c} \sum_{i \in \mathcal{P}_c} l_i(w). \tag{1}$$

In the equation, C represents the number of separate clients, where $\mathcal{P}_c$ is the set of indexes of the data points from the client c, with $p_c = |\mathcal{P}_c|$. $l(w)$ denotes the global loss function, and $L_c(w)$ denotes the local objective function. In the tth communication round, the model of the individual client is represented as w_c^t, and the processing formula of the parameter updates can be defined as follows:

$$\Delta w_c^{t+1} = f(w^t, D_c) - w^t, \quad c = 1, \ldots, C \tag{2}$$

where $f(\cdot)$ denotes stochastic gradient descent (SGD) [29] and D_c denotes the local data of client c. After uploading, the server collects the updates from each device and aggregates the model through averaging. This process can be defined as follows:

$$w^{t+1} = w^t + \sum_{i=c}^{C} \frac{D_c}{D} \Delta w_c^{t+1} \tag{3}$$

where w^t denotes the global model of the tth round.

This popular distributed baseline [6] considered the aggregation process as the averaging of the weights of each client. By introducing the federated SGD algorithm [30], it reduces the communication rounds by raising the local training epoch. Furthermore, in this concept, Li et al. [22] used convergence guarantees to tackle the non-IID problem and a constraint strategy to allow different devices to perform variable amounts of missions. Furthermore, in this study, Hanzely and Richtárik [23] proposed a method to clarify the role of local steps, which allow each device to learn from its private data without any communication.

In recent years, personalization of the MPL framework becomes essential to handle the heterogeneity challenges [31], [32], [33], [34]. To deal with the model heterogeneity problem, Li and Wang [24] used transfer learning and knowledge distillation to generate a global framework. In this thought, each individual device designs its own unique model using both private data and a public dataset. Mansour et al. [25] have taken into account user clustering, data interpolation, and model interpolation to conduct a systematic learning-theoretic study. To handle the data heterogeneity problem that data distribution varies greatly between devices, Arivazhagan et al. [26] proposed a method by employing a base+personalization layer thought, where the base layers are trained centrally and the personalized layers are trained locally. Fallah et al. [27] employed meta-learning [35] into MPL called PerAvg. This method aimed at finding a shared model, which performs well after each user training with its own loss function. To handle the heterogeneity of the underlying data distribution, PerAvg studied the personalized variant to easily let the current devices or the new devices adapt to their own data. A recent method provided by Dinh et al. [28] utilized Moreau envelopes [36] as the client-regularized objective function, which can decouple the binary MPL problem and substantially speed up the convex problems.

The related literature is summarized in Table I, including whether it is a personalized framework, the distribution of data, the ability to handle heterogeneous data issues, the ability to handle heterogeneous model issues, and the underlying methods.

B. SPP Networks

In the traditional deep neural networks (DNNs), the data for CNNs training and testing often need a pretreatment process: the fixed size of the input. This situation limits the rate of aspect as well as the scale of images. A well-known methodology called SPP [14] can eliminate this fixed-size limitation. As spatial pooling can maintain spatial information by pooling in local spatial bins, the algorithm add an SPP layer between convolutional layers [37] and FC layers to unify the size of the feature map. For example, the last feature map of the CNN layer is $16 \times 16 \times d$ as the dimension matrix, where d is the number of filters. The map will then be max-pooled through three scales: $4 \times 4 \times d$, $2 \times 2 \times d$, and $1 \times 1 \times d$, and the output of the SPP layer is concatenated and flatten to 21-D vector as the fixed-length representation. With this method, the data can be of arbitrary sizes, while the input does not require additional process for data cropping [15] or warping [17].

C. Feature Pyramid Networks

In the image processing field, pyramid structures are devised for semantic segmentation [38]. The pyramid structures provide the features with multiple scales [39]. In this article, focusing on the model heterogeneity problem, providing multiscale features under one infrastructure is the main idea, and a popular DNN method called FPN [19] gives us great inspiration. FPN enhances a standard convolutional network with a top-down pathway and horizontal connections, so the network can efficiently construct a rich, multiscale feature pyramid from a single resolution input images. Each layer of the pyramid can be used for detecting objects at a different scale, and each feature map utilizes the same model infrastructure, while marginal extra cost can be set for different detecting levels. Some other algorithms make more explorations. Zhao et al. [39] introduced a method called pyramid scene parsing network (PSPNet) to produce better quality results for pixel-level prediction. Liu et al. [40] exploited an adaptive feature pooling to make useful information of all feature levels.

III. Methodology

In this section, the proposed algorithm will be introduced in detail. Before the description, we will make a statement of the personalized MPL problem and the main definition considered in this article. Our method is declared in four parts. First, the overview framework of the heterogeneous MPL (HMPL) is presented. Then, a pyramidal generation method for the individual model will be discussed. Afterward, the integration

TABLE I
RELATED LITERATURE

Algorithms	Personalization	Data Distribution	Data Heterogeneity	Model heterogeneity	Method
FedAvg [6]		Private			Model Average
FedProx [22]		Private		✔	Constraint Strategy
L2SGD [23]		Private			SGD Variants
FedMD [24]	✔	Private + Global	✔	✔	Knowledge Distillation
HypCluster [25]		Private + Global	✔	✔	User Cluster
FedPer [26]	✔	Private		✔	Base + Personalization Layer
PerAvg [27]	✔	Private		✔	Meta Learning
pFedMe [28]	✔	Private		✔	Constraint Strategy
HMPL	✔	Private	✔	✔	Pyramidal Aggregation Strategy

structure for heterogeneous feature map will be stated. Finally, we will introduce a method for the layer-wise aggregation module in the central server. The notations mainly used is described in the Nomenclature.

A. Problem Statement

We target on one familiar but characteristic scenario from the daily usage of mobile devices. With the development of technology, mobile devices have greatly advanced on storage, computing, and communication. During daily usage, mobile devices produce massive data, and in this situation, major service providers expect to design an advanced artificial intelligence (AI) model for better user perception. In summary, we define this authentic scene as follows.

1) *Arbitrary Data Size:* As devices frequently produce data in personal fashion, it is common for different devices to generate data at different sizes. But, the existing MPL methods have difficulties in dealing with such sort of data distribution.
2) *Device Diversity:* In the multiparty networks, the storage, computation abilities, and communication capabilities may be different due to the diversities of hardware (CPU and memory), network connection (4G, 5G, and WiFi), and power supply.
3) *Individual-Wise Model:* Facing these situations, applying one single central model to tackle all the devices will lower the effectiveness of the individual accuracy.

B. Overview of the HMPL Framework

As is shown in Figs. 2(a) and 3, the model generation part of this framework is based on the idea of pyramid networks. The global server evaluates the equipment performances and set different models for different devices, which are named, ordinary model **(Ord Model)**, compromise model **(Com Model)**, and superior model **(Sup Model)**, which are declared in Table II. Subsequently, for better personal representation and aggregation in the downstream process, some characteristic personal layers are introduced, and this module is described in Section III-C.

Meanwhile, on the client side, to unify the various dimensions of the feature maps of customized models, we introduce a heterogeneous feature-map integration, and this special module is described in Section III-D, named SPP layer. In this module, the framework can tackle the dimension differences, which are caused either by the various data sizes or the model differences. The module location is shown in Fig. 4, and the module structure is shown in Fig. 5.

Fig. 2(b) shows the aggregation strategy on the server side. For the personalized models with different structures, we design a layer-wise model aggregation method. All the layers, including BS Layers and personal layers, follow their own aggregation rules. In particular, on the global server, the network layers with the same semantics will aggregate with each other. This method can efficiently provide aggregation solutions for customized models, and this module is described in detail in Section III-E.

C. Multiparty Pyramidal Generation

We propose a pyramidal strategy as the generation part of the framework. The model initialization is shown in Fig. 3. Taking the visual geometry group network (VGG-Net) [41] as an example, which is shown in Fig. 4, the feature output of each convolutional (ConV) layer after activation is denoted as $\{L_1, L_2, L_3\}$. The output of the final ConV layer is followed by a three-layer FC structure. In the first communication round after generating and training the initial model, the server collects the model and the running time of each device. Then, the server set two time thresholds, and according to this, the clients will be divided into three parts. The ratio is set as 3:4:3, where 30% of the clients are regarded as weak performance and 30% are high performance ones. For the 40% clients whose time consumption is between the two thresholds, the model structure will remain exactly as the initial form. This part of model is called **(Ord Model)** [Fig. 4(a)].

1) Com Model Generation Strategy [Fig. 4(b)]: Concentrating on the weak performance devices, the server issues an instruction to shrink the model. Specifically, the client is ordered to delete the last ConV layer (e.g., L_3). This operation will reduce the calculation consumption, but it will lead the convolutional part to get a compromised semantic feature map. Afterward, this feature map will be transferred to the FC layers for classification. This form of the model is named **(Com Model)**.

But, during this process, there comes about one interesting affair. The last channel scale of the feature map in the Com

TABLE II
DEFINITIONS OF SPECIAL MODELS AND LAYERS

Name	Abbreviation	Defination
Ordinary Model	**Ord Model**	Models generated on most common devices, the basis for generating all device models.
Compromise Model	**Com Model**	Personalized models generated on **weak** performing devices, compromised pruned based on the ordinary model.
Superior Model	**Sup Model**	Personalized models generated on **high** performance devices, extended based on the ordinary model.
Basic Layer	**BS Layer**	**Common** network layers included in all models.
Collaborative Personal Layer	**CP Layer**	Special network layers included in the **superior** model to generate high-level semantic features.
Isolaterd Personal Layer	**IP Layer**	Special network layers included in the **compromise** and **superior** model to unify interlayer sizes.

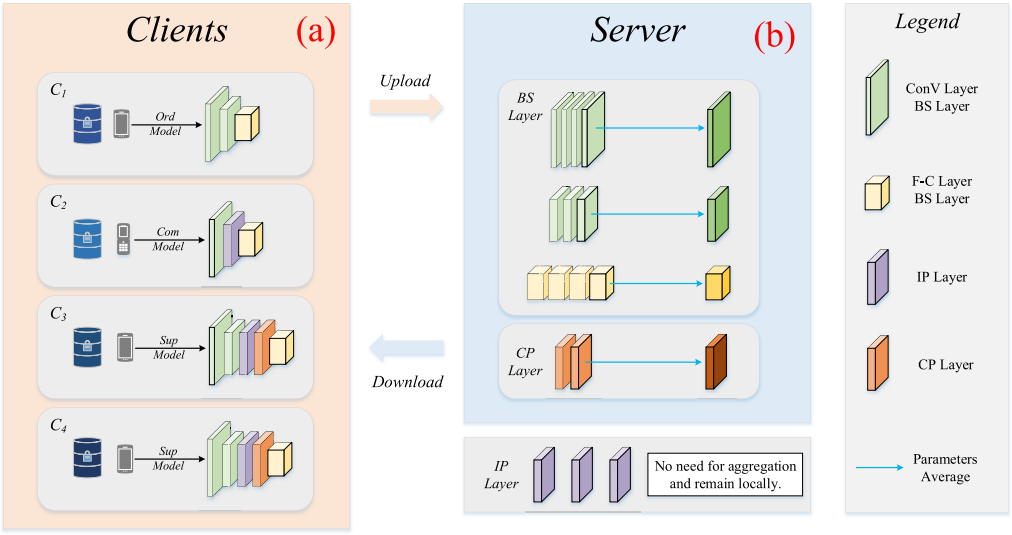


Fig. 2. Overall framework of our proposed approach. The client side shows the generation method based on the pyramid network, and this module will be discussed in Section III-C. The different models are named the *Ord Model*, *Com Model*, and *Sup Model*. The server side shows the layer-wise aggregation strategy, and this module will be discussed in Section III-E. All the customized models are made up of the *BS Layer*, *CP Layer*, and *IP Layer*. The network layers with the same semantics will aggregate with each other. (a) Clients. (b) Server.

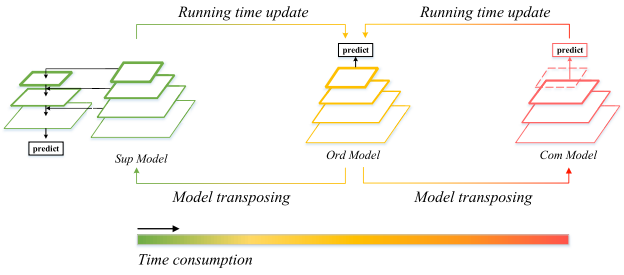


Fig. 3. Pyramid distribution strategy in the model generation process. The model in green represents the *Sup Model*, the model in yellow represents the *Ord Model*, and the model in red is *Com Model*.

Model may be different, since L_3 and L_2 may generate different channel numbers. In this situation, a 1×1 ConV layer is added to align the channel numbers. This personal ConV layer belongs to an *IP Layer*.

2) Sup Model Generation Strategy [Fig. 4(c)]: Focusing on the powerful devices whose circulated time is less than the bound time threshold, the server issues a distinct instruction. We introduce a top-down architecture with lateral connections to build high-level semantic feature maps. During this strategy, marginal extra costs are considered for stronger feature maps. This set of devices will receive the instruction of upgrading the original model into an FPN model.

First, the last pair of neighbor convolutional layers are selected (e.g., L_3 and L_2), the outputs of which are unified into the same channel dimension and then merged. In this part, a 2×2 upsampling layer and a 1×1 ConV layer are introduced for aligning. After merging, a 3×3 ConV layer is added on the last to generate the final feature map. This transform will be iterated a few communication rounds until all the convolutional layers are considered. If the circulated time exceeds the upper time threshold, it means the extra structure

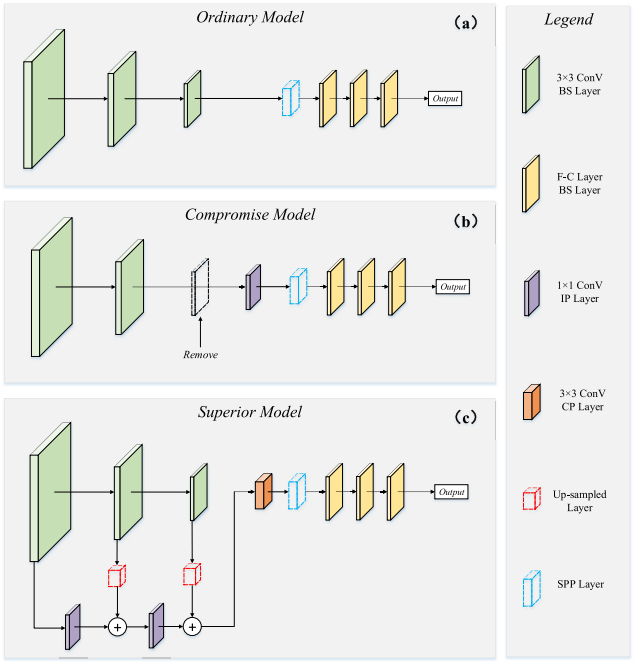


Fig. 4. Internal structure of (a) *Ord Model*, (b) *Com Model*, and (c) *Sup Model*. The definition of each layer is shown in the legend module. The network layer of the dashed line means that this part does not add additional network parameters.

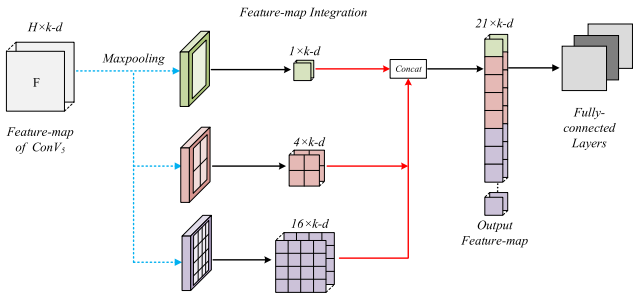


Fig. 5. Structure of the SPP layer. The blue dashed lines indicate max-pooling layer, and the red lines denote the concatenate operation.

is too large, and the model complexity is beyond tolerance. During this time, the model will stop expansion. This form of model is named **(Sup Model)**.

One very noteworthy point is that, in both Com Model and Sup Model, there generates extra 1×1 ConV layers for aligning. This part of layers are defined as *IP Layer*. For Sup Model, the final extra 3×3 ConV layer is defined as *collaborative personal (CP Layer)*. Since all the customized models are generated from the Ord Model, we define each layer of the Ord Model as *BS Layer*.

The differences between two sets of personal layers are that, all the CP Layers represent the same semantic information, while each IP Layer represents completely different semantic information from each other. The layer-wise aggregation strategy will be described in Section III-D.

D. Heterogeneous Feature-Map Integration

Through the treatments of Section III-C, all the final feature maps are aligned as the same channels. But, different structures and data sizes will lead the feature maps to different aspect ratios. During the model generation part, it can tell that all the personal models are generated by an original model, and the task-orientation layers are coincident. Such situation may not be a problem during the individual-wise training, but it will become a great issue when aggregation. Under this situation, we start to consider the inconsistent size problem, which is the feature maps need to become fixed length before transmitting to FC layers.

To adopt this request, an SPP layer is employed. Fig. 5 shows the structure of the SPP layer. Suppose the feature map of all the structures is aligned to $n \times n \times 256$, and it will be max-pooled in three ways and then concatenated. First, the feature map will be max-pooled into the size of $1 \times 1 \times 256$. In parallelization by adaptive max pooling, it will also be pooled into the size of $2 \times 2 \times 256$ and $4 \times 4 \times 256$, respectively. The results of these three pooling features will concatenate together as a global feature map with the size of 21×256 as the output of the specific pooling.

In MPL scenario, the advantages of SPP are that, it can be deployed on the client side without adding additional network parameters. This will not incur any additional communication costs. In addition, the SPP layer is capable of using the same network architecture to handle data with any aspect ratio. Moreover, it also effectively addresses the issue of inconsistent feature map sizes caused by a variety of image sizes or differences in model structures.

E. Layer-Wise Model Aggregation

Finally, we deal with the aggregation problem. Before discussion, we first redeclare three forms of the layers. All the layers of the Ord Model are named *BS Layer*. The other two personal layers named *CP Layer* and *IP Layer* are described in Section III-C and illustrated in Fig. 4. Thus, *BS Layer*, *CP Layer*, and *IP Layer* make up all the device models, and each model includes at least one part.

The set of BS Layer parameters in client c can be denoted as $\mathbf{W}_{\mathbf{c}}^{\mathrm{BS}} = (\boldsymbol{w}_{c,1}^{\mathrm{BS}}, \boldsymbol{w}_{c,2}^{\mathrm{BS}}, \ldots, \boldsymbol{w}_{c,K_B}^{\mathrm{BS}})$, where K_B is the total number of layers. Meanwhile, the set of IP Layer of client c is denoted as $\mathbf{W}_{\mathbf{c}}^{\mathrm{IP}} = (\boldsymbol{w}_{c,1}^{\mathrm{IP}}, \boldsymbol{w}_{c,2}^{\mathrm{IP}}, \ldots, \boldsymbol{w}_{c,K_I}^{\mathrm{IP}})$, where K_I is the total number of the IP Layer. Significantly, only the Sup Model has one CP Layer, and the CP Layer of client c is denoted as $\boldsymbol{w}_c^{\mathrm{CP}}$. The set of CP Layer can be denoted as $\mathbf{W}^{\mathrm{CP}} = (\boldsymbol{w}_1^{\mathrm{CP}}, \boldsymbol{w}_2^{\mathrm{CP}}, \ldots, \boldsymbol{w}_C^{\mathrm{CP}})$, where C is the total number of the clients. If the client c does not have CP Layers, $\boldsymbol{w}_c^{\mathrm{CP}}$ will be set as 0.

In the first few rounds of communication, the server will continuously monitor the communication time and issue instructions for the most suitable customized model. After all the models are established, the server will collect a certain number of heterogeneous models for interaction. The aggregation method on the central server can be summarized as the layer-wise aggregation strategy, which is shown in Fig. 2. The general gradient descent of client c in communication round t is defined as follows:

$$\Delta(\boldsymbol{w}_c) = E_{x \sim D_c}\left[\Delta_{\boldsymbol{w}}\left(f_c\left(x, y; \mathbf{W}_{\mathbf{c}}^{\mathrm{BS},\mathbf{t}}, \boldsymbol{w}_{\boldsymbol{c}}^{\boldsymbol{CP},\boldsymbol{t}}, \mathbf{W}_{\mathbf{c}}^{\mathrm{IP},\mathbf{t}}\right)\right)\right] \quad (4)$$

where $f_c(w_c, D_c) = l(x_c, y_c; w_c)$. For a machine learning problem, f_c represents the loss of the prediction on example (x_c, y_c) with model parameters w_c.

Then, we focus on the aggregation process of each layer. As $\mathbf{W}^{\mathrm{BS}}$ is the foundation of the modeling process, this part of the layers is semantically identical. Thus, all the BS Layers in every model are aggregated and upgraded as normal average method, which can be denoted as follows:

$$\Delta \mathbf{W}_{\mathbf{c}}^{\mathrm{BS,t+1}} = f_c\left(\mathbf{W}_{\mathbf{c}}^{\mathrm{BS,t}}, D_c\right) - \mathbf{W}^{\mathrm{BS,t}} \tag{5}$$

$$\mathbf{W}^{\mathrm{BS,t+1}} = \mathbf{W}^{\mathrm{BS,t}} + \eta \sum_{c=0}^{C} \frac{D_i}{D} \Delta \mathbf{W}_{\mathbf{c}}^{\mathrm{BS,t+1}}. \tag{6}$$

It is worth noting that in the Com Model, some ConV layers have been removed for compromise purposes. These part of deleted layers will not participate in the aggregation calculation.

Focusing on the CP Layers, only the Sup Models have one layer of this structure, so they will only aggregate and average with each other, which can be defined mathematically as follows:

$$\Delta \boldsymbol{w}_{\boldsymbol{c}}^{\boldsymbol{CP,t+1}} = f_c\left(\boldsymbol{w}_{\boldsymbol{c}}^{\boldsymbol{CP,t}}, D_c\right) - \boldsymbol{w}^{\boldsymbol{CP,t}} \tag{7}$$

$$\boldsymbol{w}^{\boldsymbol{CP,t+1}} = \boldsymbol{w}^{\boldsymbol{CP,t}} + \eta \sum_{c=0}^{S} \frac{D_i}{D} \Delta \boldsymbol{w}_{\boldsymbol{c}}^{\boldsymbol{CP,t+1}} \tag{8}$$

where S is the total number of the Sup Model.

Finally, since all the IP Layers represent different semantics, some defining as aligning output features and some defining upsampling, aggregating would not be sensible. These individual layers will be kept locally.

IV. Experiments

First, the popular datasets used in this article are described. Several personalized MPL algorithms as well as the classical MPL methods are introduced. Afterward, the hyperparameters and the architecture of each network will be described in detail. The effectiveness and scalability will be then studied toward comparison methods to validate the proposed approach. Finally, some ablation studies are deliberated to analyze the parameter validity.

A. Datasets and Comparison Algorithms

In this work, as one of the main destination is treating the arbitrary picture sizes, we evaluate the model on image classification tasks. We compare our method with SOTA methods on four datasets: Modified National Institute of Standards and Technology (MNIST), Fashion MNIST, Cifar10, and Cifar100.

MNIST [42]: It is a popular dataset containing 70 000 handwritten digits. It is separated into ten classes represented as number 0–9. It consists of 60 000 training and 10 000 testing images. Each sample is made up as a 28 × 28 gray scale with single channel because of the anti-aliasing techniques.

Fashion MNIST [43]: It is a new dataset for the classification task. It is also associated with one label from ten classes based on the clothing category, such as *t-shirts*, *sneakers*, and *bag*. Similar to MNIST, it has the exact same scale of 60 000 training images and 10 000 testing images. Each sample is centered in a 28 × 28 gray scale with one channel. We call this FAMNIST in the following for simplicity.

Cifar10 and Cifar100 [44]: They consist of 60 000 color images, and each image has 32 × 32 pixels with three channels. Cifar10 is a ten-way classification problem with 6000 images per class. There are 5000 samples for training and 1000 for testing. Cifar100 is a 100-way classification problem containing 600 images in each class, where 500 are selected for training and 100 for testing.

We compare our method with the classical approach named FedAvg and seven popular approaches named FedProx, federated learning via model distillation (FedMD), FedPer, PerAvg, and pFedMe.

FedAvg [6]: It is one of the fundamental methods on MPL task. It employs the gradient averaging method to simplify an aggregation process. By employing FedSGD, FedAvg raises the local training epoch while decreasing the communication round to reduce the costs.

FedProx [22]: It tackles the system heterogeneity of significant variability and the non-IID data across the network. It utilizes convergence guarantees and a constraint strategy to allow different devices to perform variable amounts of mission.

Loopless local stochastic gradient descent (L2SGD) [23]: It proposes a method to clarify the role of local steps, which allow each device to learn from its private data without any communication.

FedMD [24]: It uses transfer learning and knowledge distillation to handle model heterogeneity problem. By transferring MNIST–FAMNIST pair and Cifar10–Cifar100 pair, it enables devices to own uniquely designed models.

HypCluster [25]: It has taken into account user clustering, data interpolation, and model interpolation to conduct a systematic learning-theoretic study.

FedPer [26]: It proposes a base + personalization layer method. By sharing the BS Layer while keeping the personal layer locally, it can combat the ill effects of statistical heterogeneity.

PerAvg [27]: It introduces meta-learning into MPL to solve the personalized problem. With studying a personalized variant, the destination of this method is to find an initial shared model for all the users to adapt their local dataset.

pFedMe [28]: It uses Moreau envelopes as the regularization loss for each client, which can help decouple the optimization process of personalized model from global server.

B. Parameter Settings

For all the algorithms, the label space and the feature space of all the datasets are divided into 1000 per device. In each round, ten devices are selected for communication. In this work, since we do not consider the non-IID problem, all the labels are allocated uniformly. To simulate the scene of arbitrary data sizes, we randomly crop each image in all the datasets within the set [(16:9), (4:3), (1:1), (3:4), (9:16)]. To simulate the differentials in equipment performances, the devices are deployed in three experimental environments with different configurations. As is shown in Table III, the devices are divided by 3:4:3 with slight perturbation.

TABLE III
EXPERIMENTAL ENVIRONMENTS

Performance	GPU	RAM	Proportion
Ord Model	RTX 2080Ti	11019MiB	40%(±10%)
Com Model	Tesla P40	22919MiB	30%(±10%)
Sup Model	RTX 3090	24576MiB	30%(±10%)

The proposed approach and the comparison algorithms are using the same cases of the parameter settings, where the subset of clients is fixed as $S = 10$, and the learning rates adaptively change from 0.01 to 0.001 during the communication rounds. For MNIST and FAMNIST datasets, CNNs of our Ord Model and all the methods contain three 3×3 kernel filters with (64, 128, and 256) channels, respectively. Each convolutional layer is connected by a 2×2 max-pooling layer, and two FC layers with 1024 neurons and one ten-class softmax classification layer are followed afterward. The batch size is set as 20, and the number of communication rounds is set as 800 while introducing early stopping. For Cifar10 and Cifar100 datasets, the structure of CNNs can be described as (64, pooling; 128, pooling; and 384, 256, and 256, pooling), where the digits are represented as the channel number of the 3×3 kernel filters and the word "*pooling*" represents the 2×2 max-pooling layer. The CNN structure is followed by a three-layer MLP classifier with (4096, 1024, and 10) neurons. The batch size is set as 64, and the number of communication rounds is set as 1600 while introducing early stopping. The training and testing ratios are divided by the datasets without modification.

Other hyperparameters are set by the best performance of each comparison methods. For FedProx, we set the local epoch $E = 1$ and the regularization parameter $\mu = 0.1$. For FedMD, the structure of each individual device is initialized in full accordance with the supplied parameters, and the weight parameter c_k is set as 0.2. For FedPer, the level of statistical heterogeneity k and the aggregation rounds for different layers K_P are set as the best performance according to the different datasets, and all the models are generated as the fine-tune form. For PerAvg, we use its personalized model, which can be denoted as the local model that takes an SGD step from the global server, and the second learning rate is set as $\beta = 0.005$. For pFedMe, we set the computation steps $K = 5$, the average moving parameter $\beta = 1$, and the strength parameter $\lambda = 20$, which is proven the best. Experiments of each approach are repeated five times and report the average.

C. Comparison With the State of the Art

For personalized approaches, as the backbones have the ability to handle the arbitrary image sizes, the input data are operated as the same form of our proposed method. The experimental results are shown in Table IV. Moreover, for FedAvg, FedProx, and L2SGD, since the algorithms have difficulties on treating the arbitrary sizes, the data remain undisposed. To compare with these methods, our algorithm also applies the original data, and the results are shown in Table V. In the table, the differences between HMPL and HMPL$_S$ are that, HMPL is the complete framework, and HMPL$_S$ only employs the heterogeneous feature-map integration method in Section III-C while shielding other modules.

First, we analyze the personalized comparisons in Table IV. Our proposed method obtains almost the best results on all the datasets. As FedMD uses transfer learning and knowledged distillation, the advantage knowledge of MNIST and Cifar10 directly migrates to FAMNIST and Cifar100. This leads to performance improvements on two more complex datasets and achieves the best result on Cifar100. PerAvg, which introduces the idea of meta-learning to MPL, acquires less good performance. But, since the results have the minimum variance, the method performs the most stable structure. For pFedMe, the approach provides the minimum loss whose convergence benefits from the Moreau envelope for regularization. Our method improves the SOTA of all the datasets. For Cifar10, on arbitrary data form, our method achieves a 12.3% improvement.

Then, for the regular images, comparing with three classical methods, our approach gains moderate lift. Due to the better information integrality, the results of this group are generally better than the previous one. For Cifar10 and Cifar100 datasets, our approach gains the improvements by 5.8% and 4.9%. For HMPL$_S$, the results have a certain degree of improvement comparing with FedAvg. When contrasting HMPL$_S$ to HMPL, the effect has been slightly improved. But, the disadvantage is that, this part of promotion leads to extra consumption on both time and memory.

D. Ablation Study

In this section, we will discuss the influences and the effects of the three generation structures. For discussing the arbitrary sizes of data, as FedAvg have difficulties on aggregation, we add the SPP layer on the end of the ConV layer for comparison. In addition, we also discuss the case of original size for integrity.

Fig. 6 represents the test accuracy and training loss of arbitrary image sizes. Since each image is randomly clipped, some important information about features may be lost. In this situation, the Sup Model and the Ord Model both gain better performance against the FedAvg approach. The reason is that, the superior feature has advantages to driving the model aggregation and promoting the presentation of the global framework. In other words, the additional superior structure can also help the global structure alleviate the impact of compromise strategies, and provide prior knowledge for the Com Model. Focusing on the results of each separate device, the shadow represents the variances of ten devices. In this case, the Ord Model has a lower training loss due to the additional structure of the Sup Model resulting in larger losses. As all the accuracies are close to 100%, this part of outcomes has not much difference.

Fig. 7 shows the results of the original sizes of data. In this part, the extra traditional FedAvg is considered. The other parts are as the same meaning as above. The figure illustrates that our strategy also outperforms on normal sizes of data.

TABLE IV
COMPARISON OF BASELINES ON ARBITRARY IMAGE SIZES

Algorithm	Dataset							
	MNIST		FAMNIST		Cifar10		Cifar100	
	Acc	Loss	Acc	Loss	Acc	Loss	Acc	Loss
FedMD	\	\	0.803(±0.0429)	0.582(±0.0271)	\	\	**0.509(±0.0372)**	**2.076(±0.6258)**
HypCluster	0.954(±0.0007)	0.193(±0.0059)	0.851(±0.0607)	0.659(±0.0981)	0.801(±0.0003)	0.904(±0.5467)	0.457(±0.0048)	5.802(±1.1275)
FedPer	0.935(±0.0351)	0.249(±0.1080)	0.762(±0.0607)	1.147(±0.4136)	0.714(±0.0031)	1.973(±0.1172)	0.250(±0.0660)	9.774(±0.4936)
PerAvg	0.931(±0.0001)	0.292(±0.0001)	0.732(±0.0002)	0.826(±0.0025)	0.658(±0.0007)	2.175(±0.3734)	0.156(±0.0016)	3.710(±0.5604)
pFedMe	0.979(±0.0002)	0.052(±0.0011)	0.878(±0.0001)	**0.148(±0.0163)**	0.710(±0.0043)	0.872(±0.8303)	0.289(±0.0033)	2.961(±0.8333)
HMPL$_S$	0.982(±0.0002)	0.065(±0.0026)	0.857(±0.0002)	0.421(±0.0175)	0.828(±0.0009)	0.767(±0.9728)	0.436(±0.0021)	3.853(±0.6200)
HMPL	**0.986(±0.0001)**	**0.047(±0.0017)**	**0.886(±0.0003)**	0.314(±0.0160)	**0.837(±0.0001)**	**0.637(±0.0135)**	0.502(±0.0016)	2.876(±0.0317)

TABLE V
COMPARISON OF BASELINES ON REGULAR IMAGE SIZES

Algorithm	Dataset							
	MNIST		FAMNIST		Cifar10		Cifar100	
	Acc	Loss	Acc	Loss	Acc	Loss	Acc	Loss
FedAvg	0.978(±0.0014)	0.043(±0.0013)	0.879(±0.004)	0.060(±0.0013)	0.799(±0.0039)	0.718(±0.0229)	0.408(±0.0027)	2.322(±0.4592)
FedProx	0.979(±0.0006)	**0.043(±0.0009)**	0.874(±0.0005)	0.072(±0.0014)	0.816(±0.0024)	0.512(±0.0218)	0.487(±0.0031)	1.955(±0.3081)
L2SGD	0.982(±0.0008)	0.050(±0.0010)	0.880(±0.0010)	0.065(±0.0009)	0.805(±0.0017)	**0.493(±0.0052)**	0.451(±0.0132)	2.757(±0.7033)
HMPL$_S$	0.983(±0.0001)	0.062(±0.0029)	0.876(±0.0030)	**0.032(±0.0014)**	0.851(±0.0090)	0.781(±0.0166)	0.494(±0.0035)	3.643(±0.8050)
HMPL	**0.987(±0.0006)**	0.045(±0.0007)	**0.889(±0.0011)**	0.037(±0.0035)	**0.874(±0.0231)**	0.505(±0.0069)	**0.536(±0.0017)**	**1.684(±0.0334)**

For the test accuracy on the server, the Sup Model gains the best performance, while the FedAvg with SPP layer obtains better performance than the Ord Model. This is because the SPP layer can provide more location features as well as semantic features, and this facilitates the FC layer for classification. For the accuracies and losses on the client scenario, such as the previous experiments, all the curves have similar results. In this part, the Sup Model and the Ord Model have slight increase. The result is that, on the normal image scene, the superior strategy can also advance all the device model through aggregation and communication.

E. Influence of Different Distribution Ratios

In this section, we are curious about which ratio of the distribution is the best. Thus, we make a grid search [45] from 10% to 50% of the Com Model and from 40% to 60% of the Ord Model. The remaining proportion represents the Sup Model. Fig. 8 illustrates the testing results of each model under different ratios. Before learning, we thought more Sup Models would bring better performances, but the experimental results show that, the balanced proportion of each part produces the best performance instead. The reason is that, if any structure has been distributed with a low ratio, the personal aggregation will be inadequate. This part of the model will take a restricted view of information into account and produces negative impacts on the global model.

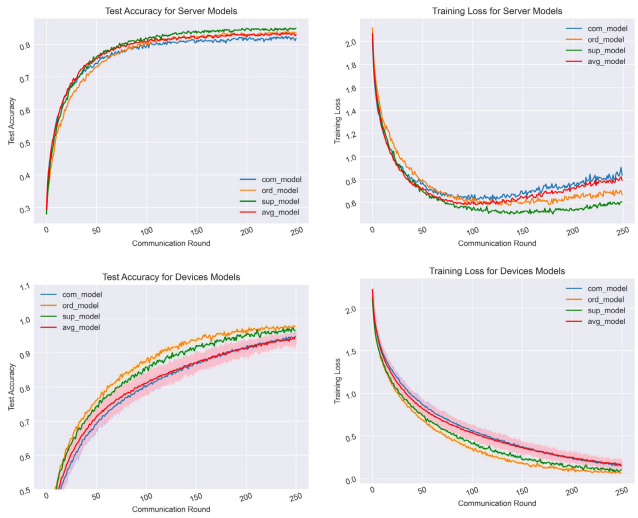


Fig. 6. Test accuracy and the training loss of arbitrary image sizes on both server side and device side. The shadow represents the variance of the performance from separate devices.

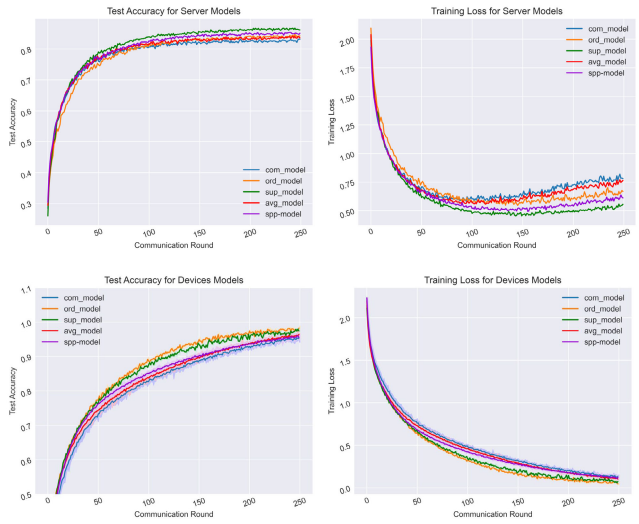


Fig. 7. Test accuracy and the training loss of regular image sizes on both server side and device side. The shadow represents the variance of the performance from separate devices.

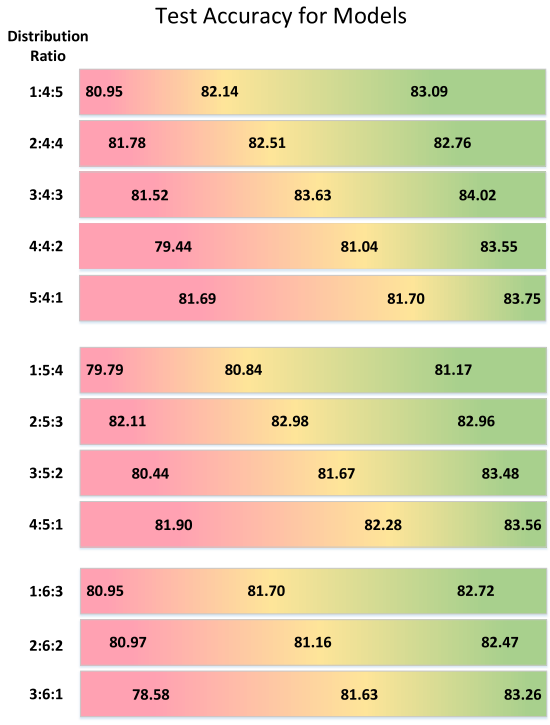


Fig. 8. Test accuracy for various distribution proportions of our algorithm. The red part represents the ratio of the *Com Model*, the yellow part represents the *Ord Model*, and the green part is the *Sup Model*.

Fig. 9 shows the grid-search results of three parts of the models. It is important to note that, when each part is close to the mid-value, the global model performs the best. When the ratio of the Com, Ord, and Sup Models is 3:4:3, the accuracy of the global model can achieve 85.44%. As the proportion changes, the accuracy decreases. But, when the ratio becomes 2:6:2, the results have a certain recovery. The results of this phenomenon will be discussed in our following work.

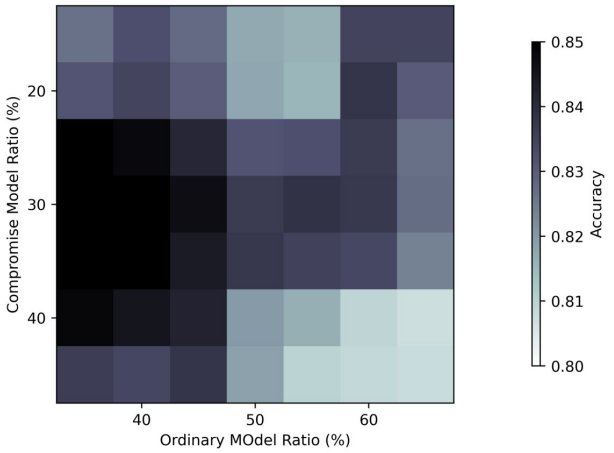


Fig. 9. Accuracy of the global server by using different distribution ratios.

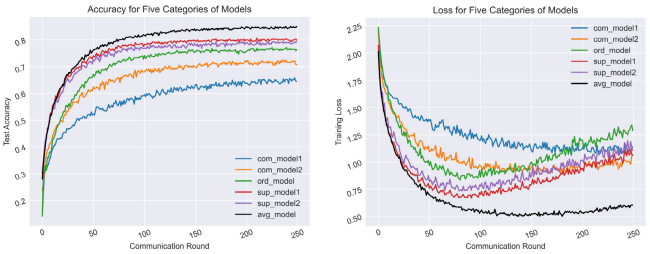


Fig. 10. Results of five forms of the model by generation operation. The black line represents the FedAvg algorithm, and the rest of the lines represent the performances of each model.

F. Condition of Five Various Models

In this chapter, we will analyze what results will be obtained if the personalized models are generated by five generators. We separate the Com Model and the Sup Model into two parts, respectively, named *com_model1*, *com_model2*, *sup_model1*, and *sup_model2*. Based on com_model2, com_model1 further simplifies the model, while sup_model2 adds an additional structure to the FPN. Each part of the model is distributed at 20%. The results are shown in Fig. 10.

In the diagram, we can see that the performance of the framework becomes highly unstable. On the 100th communication round, the Com Models have not yet converged, while the Ord and the Sup Models are starting overfitting. The loss curves of these three structures continue to raise until the end. Due to the model becoming too scattered and the insufficiencies of communication, the performances of each model are worse than FedAvg. In this situation, the Sup Models fail to improve the global performance, while the Com Models play a nonignorable misleading role in the overall results. In some com_model2 structures, there may occur a situation where the ConV structure may only exist one ConV layer for blindly simplifying the model. This will lead the global negatively optimized.

G. Visualization of Various Models

In this section, we will discuss the visualization results. First, we discuss the evolution trend of server testing among different approaches, and then, the device views training

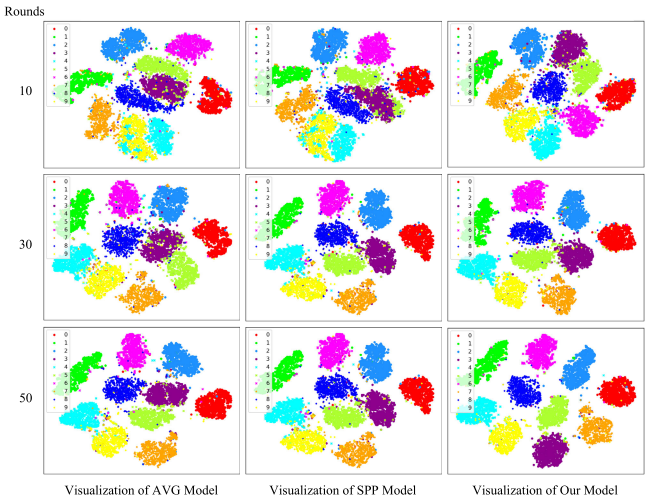


Fig. 11. Changing trend of visualization results on the global server; 10 000 nodes (all nodes) of the testing set are selected from the MNIST dataset. The numerical representation is shown in the legend.

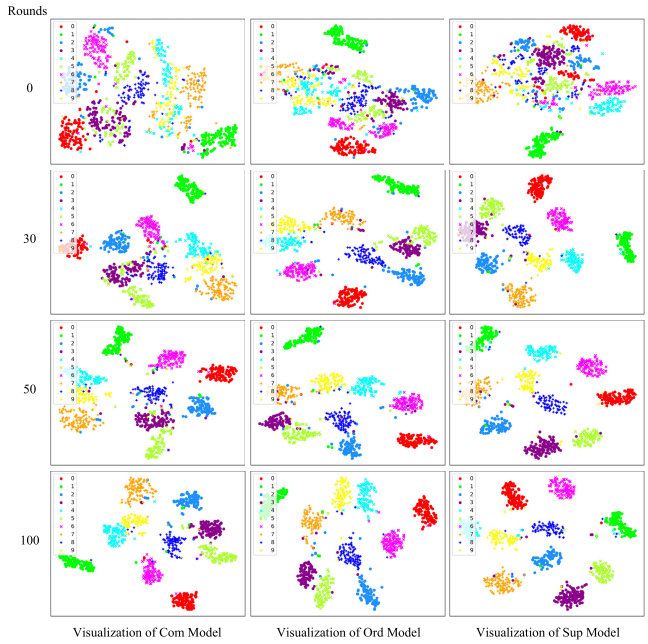


Fig. 12. Changing trend of visualization results on the local server; 1000 nodes of the training set are selected from the MNIST dataset. The numerical representation is shown in the legend.

among different generate strategies. The sketch maps contain 10 000 nodes for the server and 1000 nodes for the devices. We utilize the t-distributed stochastic neighbor embedding (t-SNE) [46] as the visualization method and use the MNIST dataset for training.

As is shown in Fig. 11, the contrasts are described among the FedAvg model, the SPP model, and the HMPL model at 10–50 rounds. After ten rounds of communication, FedAvg and SPP still have trouble distinguishing numbers 4 and 9, and number 5 also has incomplete clustering, while HMPL has a preliminary clustering effect. At 30 rounds of communication, SPP can gain a comparable effect toward HMPL, while the clustering radius of FedAvg is still large. After 50 rounds of communication, HMPL obtains sufficient dispersion clustering centers and the smallest cluster radii.

In Fig. 12, the sketch maps show the visualization results of HMPL of one device using different generating strategies. From the initial states to the results after 100 rounds, the Sup Model shows the best performance. At 30 rounds of communication, the Ord Model has a preliminary classification effect, while the Com Model is still not obvious, and the Sup Model has already achieved a visible effect comparable with the Com Model at 100 rounds. From the last line of the map, we can see that after 100 rounds, the Sup Model can reach an accuracy of more than 99%, and the clustering centers of each class have best dispersion effect.

V. Conclusion

In this article, we tackle two issues of practical significance through MPL. First, facing the problem of various devices holding arbitrary sizes of data, we introduce a heterogeneous feature-map integration method. This method provides a good solution to the limitations on the client side. Second, facing model heterogeneity and statistical heterogeneity, we propose a layer-wise model generation and aggregation strategy. Such a strategy can maximize the calculation power of advanced equipment while taking care of weak devices through the personal generation and similar semantic aggregation. Extensive experiments are conducted on four popular datasets with arbitrary size form and regular size form, and the result demonstrates that our proposed model outperforms the SOTA. In the future, we will explore more practical problems in personal MPL and provide solutions.

References

[1] M. N. H. Nguyen et al., "Self-organizing democratized learning: Toward large-scale distributed learning systems," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, May 10, 2022, doi: 10.1109/TNNLS.2022.3170872.

[2] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet Things J.*, vol. 3, no. 5, pp. 637–646, Oct. 2016.

[3] M. Armbrust et al., "Above the clouds: A Berkeley view of cloud computing," Dept. EECS, Univ. California, Oakland, CA, USA, Tech. Rep. UCB/EECS-2009-28, 2009.

[4] P. Voigt and A. Von dem Bussche, *The EU General Data Protection Regulation (GDPR): A Practical Guide*, 1st ed. Cham, Switzerland: Springer, 2017.

[5] G. J. Annas, "HIPAA regulations—A new era of medical-record privacy?" *New England J. Med.*, vol. 348, no. 15, pp. 1486–1490, Apr. 2003.

[6] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. Artif. Intell. Statist.*, 2017, pp. 1273–1282.

[7] B. Ghimire and D. B. Rawat, "Recent advances on federated learning for cybersecurity and cybersecurity for federated learning for Internet of Things," *IEEE Internet Things J.*, vol. 9, no. 11, pp. 8229–8249, Jun. 2022.

[8] M. Gong, J. Feng, and Y. Xie, "Privacy-enhanced multi-party deep learning," *Neural Netw.*, vol. 121, pp. 484–496, Jan. 2020.

[9] P. Kairouz et al., "Advances and open problems in federated learning," *Found. Trends Mach. Learn.*, vol. 14, nos. 1–2, pp. 1–210, 2021.

[10] V. Kulkarni, M. Kulkarni, and A. Pant, "Survey of personalization techniques for federated learning," in *Proc. 4th World Conf. Smart Trends Syst., Secur. Sustainability (WorldS4)*, Jul. 2020, pp. 794–797.

[11] Q. Wu, K. He, and X. Chen, "Personalized federated learning for intelligent IoT applications: A cloud-edge based framework," *IEEE Open J. Comput. Soc.*, vol. 1, pp. 35–44, 2020.

[12] V. Smith, C.-K. Chiang, M. Sanjabi, and A. S. Talwalkar, "Federated multi-task learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 1–11.

[13] A. Imteaj, U. Thakker, S. Wang, J. Li, and M. H. Amini, "A survey on federated learning for resource-constrained IoT devices," *IEEE Internet Things J.*, vol. 9, no. 1, pp. 1–24, Jan. 2022.

[14] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 9, pp. 1904–1916, Sep. 2015.

[15] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 25, 2012, pp. 1–9.

[16] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 580–587.

[17] J. Donahue et al., "DeCAF: A deep convolutional activation feature for generic visual recognition," in *Proc. Int. Conf. Mach. Learn.*, 2014, pp. 647–655.

[18] Y. Chen, X. Sun, and Y. Jin, "Communication-efficient federated deep learning with layerwise asynchronous model update and temporally weighted aggregation," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 10, pp. 4229–4238, Oct. 2020.

[19] T. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 936–944.

[20] B. Varghese, N. Wang, S. Barbhuiya, P. Kilpatrick, and D. S. Nikolopoulos, "Challenges and opportunities in edge computing," in *Proc. IEEE Int. Conf. Smart Cloud (SmartCloud)*, Nov. 2016, pp. 20–26.

[21] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 2, pp. 1–19, 2019.

[22] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," *Proc. Mach. Learn. Syst.*, vol. 2, pp. 429–450, Mar. 2020.

[23] F. Hanzely and P. Richtárik, "Federated learning of a mixture of global and local models," 2020, *arXiv:2002.05516*.

[24] D. Li and J. Wang, "FedMD: Heterogenous federated learning via model distillation," 2019, *arXiv:1910.03581*.

[25] Y. Mansour, M. Mohri, J. Ro, and A. T. Suresh, "Three approaches for personalization with applications to federated learning," 2020, *arXiv:2002.10619*.

[26] M. G. Arivazhagan, V. Aggarwal, A. K. Singh, and S. Choudhary, "Federated learning with personalization layers," 2019, *arXiv:1912.00818*.

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

[28] C. T. Dinh, N. Tran, and J. Nguyen, "Personalized federated learning with Moreau envelopes," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 21394–21405.

[29] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proc. 19th Int. Conf. Comput. Statist. (COMPSTAT)*. Paris France: Springer, Aug. 2010, pp. 177–186.

[30] J. Chen, X. Pan, R. Monga, S. Bengio, and R. Jozefowicz, "Revisiting distributed synchronous SGD," 2016, *arXiv:1604.00981*.

[31] A. Z. Tan, H. Yu, L. Cui, and Q. Yang, "Towards personalized federated learning," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Mar. 28, 2022, doi: 10.1109/TNNLS.2022.3160699.

[32] Y. Jiang et al., "Model pruning enables efficient federated learning on edge devices," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Apr. 25, 2022, doi: 10.1109/TNNLS.2022.3166101.

[33] F. Sattler, K. Müller, and W. Samek, "Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 8, pp. 3710–3722, Aug. 2021.

[34] Y. Zhao et al., "Personalized federated few-shot learning," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Jul. 21, 2022, doi: 10.1109/TNNLS.2022.3190359.

[35] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 1126–1135.

[36] J.-J. Moreau, "Propriétés des applications prox," *Comptes Rendus de l'Académie des Sciences (Série A, Mathématiques)*, vol. 256, pp. 1069–1071, 1963.

[37] Y. LeCun et al., "Backpropagation applied to handwritten zip code recognition," *Neural Comput.*, vol. 1, no. 4, pp. 541–551, Dec. 1989.

[38] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 4, pp. 834–848, Apr. 2018.

[39] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 6230–6239.

[40] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, "Path aggregation network for instance segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 8759–8768.

[41] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*.

[42] L. Deng, "The MNIST database of handwritten digit images for machine learning research [best of the web]," *IEEE Signal Process Mag.*, vol. 29, no. 6, pp. 141–142, Nov. 2012.

[43] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms," 2017, *arXiv:1708.07747*.

[44] A. Krizhevsky et al., "Learning multiple layers of features from tiny images," Univ. Toronto, Toronto, ON, Canada, Tech. Rep., 2009. [Online]. Available: https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf

[45] S. M. LaValle, M. S. Branicky, and S. R. Lindemann, "On the relationship between classical grid search and probabilistic roadmaps," *Int. J. Robot. Res.*, vol. 23, nos. 7–8, pp. 673–692, Aug. 2004.

[46] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, pp. 2579–2605, Nov. 2008.

Yuanqiao Zhang received the B.S. degree in electronic engineering from Xidian University, Xi'an, China, in 2016, where he is currently pursuing the Ph.D. degree in electronic science and technology with the Key Laboratory of Collaborative Intelligence Systems, Ministry of Education.

His research interests include multiparty learning, collaborative optimization, and encrypted computing.

Maoguo Gong (Senior Member, IEEE) received the B.S. degree (Hons.) in electronic engineering and the Ph.D. degree in electronic science and technology from Xidian University, Xi'an, China, in 2003 and 2009, respectively.

Since 2006, he has been a Teacher with Xidian University. In 2008 and 2010, he was promoted as an Associate Professor and as a Full Professor, respectively, both with exceptive admission. His research interests include the area of computational intelligence with applications to optimization, learning, data mining, and image understanding.

Dr. Gong is an Associate Editor of the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION and the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS.

Yuan Gao received the B.S. degree from the School of Artificial Intelligence, Xidian University, Xi'an, China, in 2018, and the Ph.D. degree from the School of Electronic Engineering, Xidian University, in 2022.

He was a Visiting Scholar with the School of Computer Science and Engineering, Nanyang Technological University, Singapore, from 2021 to 2022. He is currently a Lecturer with the Key Laboratory of Collaborative Intelligence Systems, Ministry of Education, Xidian University. His research interests include edge computing, model fusion, and collaborative learning.

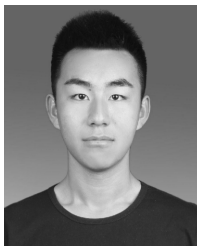
A. K. Qin (Senior Member, IEEE) received the B.Eng. degree from Southeast University, Nanjing, China, in 2001, and the Ph.D. degree from Nanyang Technology University, Singapore, in 2007.

From 2007 to 2017, he was with the University of Waterloo, Waterloo, ON, Canada; Institut National de Recherche en Informatique et en Automatique (INRIA), Grenoble Rhône-Alpes, France; and RMIT University, Melbourne, VIC, Australia. In 2017, he joined the Swinburne University of Technology, Melbourne, where he is currently a Professor and also the Director of the Swinburne Intelligent Data Analytics Laboratory, the Deputy Director of the Swinburne Space Technology and Industry Institute, and the Program Lead of the Swinburne Data Science Research Institute. His major research interests include machine learning, evolutionary computation, computer vision, remote sensing services computing, and pervasive computing.

Dr. Qin was a recipient of the 2012 IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION Outstanding Paper Award. He is the Chair of the IEEE Computational Intelligence Society (CIS) Neural Networks Technical Committee and the Vice-Chair of the IEEE CIS Emergent Technologies Task Force on "Multitask Learning and Multitask Optimization."

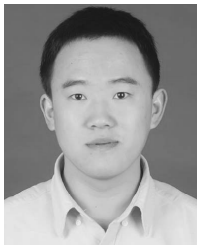
Kun Wang received the B.S. degree in computer science and technology from Xidian University, Xi'an, China, in 2018, where he is currently pursuing the Ph.D. degree in electronic science and technology with the Key Laboratory of Collaborative Intelligence Systems, Ministry of Education.

His research interests include federated learning and few-shot learning.



Yiming Lin is currently pursuing the master's degree in electronic science and technology with the Key Laboratory of Collaborative Intelligence Systems, Ministry of Education, Xidian University, Xi'an, China.

His research interests include federated learning and multimodal learning.



Shanfeng Wang (Member, IEEE) received the B.Eng. and Ph.D. degrees from Xidian University, Xi'an, China, in 2012 and 2017, respectively.

Since 2018, he has been a Teacher with Xidian University, where he is currently an Associate Professor. He has authored or coauthored more than 20 papers in refereed journals and proceedings. His research interests include computational intelligence and social network analysis.