Communication Efficient Federated Learning With Heterogeneous Structured Client Models

Yao Hu, Xiaoyan Sun, *Member, IEEE*, Ye Tian, *Member, IEEE*, Linqi Song, *Senior Member, IEEE*, and Kay Chen Tan, *Fellow, IEEE*

***Abstract*—Federated learning (FL) has recently attracted much attention due to its superior performance in privacy protection when processing data from different terminals. However, homogeneous deep learning models are pervasively adopted without considering the difference between distinct data in various clients, resulting in low learning performance and high communication costs. This paper thus proposes a novel FL framework with heterogeneous structured client models for handling different data scales and investigates its superiority over canonical FL with homogeneous models. Additionally, singular value decomposition is adopted on the client models to reduce the amount of transmitted data, i.e., the communication costs. The aggregation mechanism with multiple models on the central server is then presented based on the heterogeneous characteristics of the uploaded parameters and models. The proposed framework is applied to four benchmark classification datasets and a trend following task on electromagnetic radiation intensity time series data. Experimental results demonstrate that the proposed method can effectively improve the accuracy of local learning models and significantly reduce communication costs.**

***Index Terms*—Federated learning, heterogeneous structured model, neural network, singular value decomposition.**

Manuscript received 15 February 2022; revised 10 July 2022; accepted 26 August 2022. Date of publication 5 October 2022; date of current version 25 May 2023. This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grants 61876184, U21A20512, and 61876162, in part by the Research Grants Council of the Hong Kong, SAR under Grants PolyU11211521 and ECS 21212419, in part by Changsha Science and Technology Program International and Regional Science and Technology Cooperation Project under Grant kh2201026, in part the Technological Breakthrough Project of Science, Technology and Innovation Commission of Shenzhen Municipality under Grant JSGG20201102162000001, in part by InnoHK Initiative, the Government of the HKSAR, Laboratory for AI-Powered Financial Technologies, the Hong Kong UGC Special Virtual Teaching and Learning (VTL) Grant 6430300, in part by Hong Kong UGC RMGS under Grant 9229003, and in part by Tencent AI Lab Rhino-Bird Gift Fund. *(Corresponding authors: Xiaoyan Sun; Linqi Song.)*

Yao Hu and Linqi Song are with the Department of Computer Science, City University of Hong Kong, 999077, Hong Kong SAR, and also with the City University of Hong Kong, Shenzhen Research Institute, Shenzhen 518057, China (e-mail: yaohu4-c@my.cityu.edu.hk; linqi.song@cityu.edu.hk).

Xiaoyan Sun is with the School of Information and Control Engineering, China University of Mining and Technology, Xuzhou 221000, China (e-mail: xysun78@126.com).

Ye Tian is with the Information Materials and Intelligent Sensing Laboratory of Anhui Province, Institutes of Physical Science and Information Technology, Anhui University, Hefei 230601, China (e-mail: field910921@gmail.com).

Kay Chen Tan is with the Department of Computing, The Hong Kong Polytechnic University, 999077, Hong Kong SAR (e-mail: kctan@polyu.edu.hk).

Digital Object Identifier 10.1109/TETCI.2022.3209345

I. Introduction

IT IS widely recognized that, the data generated by different users or producers stored on various types of smart terminals constitutes a major composition of Big Data nowadays. Designing powerful machine learning methods to explore the intrinsic values of integrated data from different terminals has thus become an important research topic in Big Data [1], [2]. However, data from different users can be extremely private. Owing to the concern on privacy leakages, the users may be reluctant to share such data with others, and existing machine learning methods based on traditional data collection are difficult to work with the privacy protection [3].

Following the principle of focused collection or data minimization, Google Inc., introduced a privacy preserving machine learning framework named federated learning (FL) [4]. The idea is that on each intelligent terminal (i.e., client), a local model, especially a deep learning model, is trained to extract corporate information from many users' private data. Subsequently, the parameters of each local model, such as the weights of the deep learning networks, are uploaded to the server. On the central server, aggregation optimizations are conducted on these uploaded parameters to improve the learning performance of central and local models. Then, the aggregated parameters are transmitted to the participating clients to re-optimize the local models. Clearly, this framework can fundamentally protect users' privacy since it does not implicitly exchange or collect the user-generated data, whereas only the local model parameters are aggregated. In addition, the aggregation of model parameters can realize multi-source information fusion, which solves the cold-start problem in traditional approaches to a certain extent.

Lots of studies have already been devoted to expanding the practicability of FL. For instance, the federated averaging (FedAVG) proposed by McMahan et al. [5] reduces the training minibatch sizes or increases the number of local training epochs to ease the communication bottleneck. On the basis of FedAVG, Chen et al. [6] proposed an asynchronous learning strategy for the local models and a temporally weighted aggregation method to improve the communication efficiency along with the convergence accuracy. In comparison to traditional machine learning methods, FL focuses on protecting user's privacy. Bonawitz et al. [7] proposed a new secure aggregation protocol to process the uploaded parameters on the server only when the number of participating clients meet certain conditions, such as

a specified quantity. This strategy further reduces the possibility of data leakage caused by attacks on the server. In addition, many studies utilize FL to deal with the isolated insufficient data under the premise of privacy protection. Chen et al. [8] embedded collaborative filtering algorithm into the framework of FL for safely gaining the global social preference. All these work are generally based on the assumption that all clients adopt the homogeneous structured models, which may result in low accuracy and extra computing costs under imbalanced amount of local data.

In the framework of FL, due to the existence of heavy and light users, the number of local datasets generated and stored on various clients may be significantly different [4]. Hence, it is unreasonable to build same structural local models on each clients. Intuitively, deep models trained on small-scale datasets tend to be overfitting, while shallow models often fail to fully learn the knowledge in large-scale data. To ensure the performance of FL, this challenge should be given high attention, e.g. providing models with various structures for clients with different dataset scales. Besides that, communication bottleneck is also an unavoidable issue [9]. Since the participating clients, such as mobile phones and tablets, usually have a limited data plan with unreliable, expensive network connection, the high communication costs make some devices with small bandwidth to be absent from the parameter aggregation. As a comparison, with the development of hardware and mobile graphic processing units, these devices gradually possess powerful computation capacities and can employ complex deep neural networks [10]. However, the training update of such models consists of a large number of parameter matrices, making communication a more severe bottleneck. Generally, the reduction of communication costs deserves more attention than the decrease in computation consumption.

To this end, this paper investigates the superiority of heterogeneous structured local models under FL and proposes an efficient communication method to reduce the communication costs. Specifically, two heterogeneous structured models, i.e., the deep neural networks and shallow ones, are built as examples on the server and applied to the local clients. It is noted there have existed some public pre-designed models that satisfy the setting of deep and shallow structures, e.g., Resnet-50 and Resnet-18 [11]. These models are carefully designed by the experts can be easily adopted to new datasets by changing the number of units of final dense layers [12], [13]. Therefore, these pre-designed models can be inserted into our framework for practical applications, whose performance is validated on various application tasks [14], [15]. Then, each client selects a local model according to the amounts of its stored datasets. Considering the limited bandwidth, we subsequently propose to utilize singular value decomposition (SVD) to obtain the singular value matrices of the high-dimensional model parameters. The singular values rather than the whole model matrices are uploaded to reduce the communication costs. An aggregation method on the server is presented based on the heterogeneous characteristics of the uploaded parameters and models. The main contributions of this paper can be summarized as follows:

1) This work proposes a heterogeneous structured models FL framework and demonstrates its superiority over the FL with the homogeneous local model.
2) An SVD-based communication efficient method is proposed to reduce the amount of transmitted parameters for decreasing the communication costs.
3) The proposed methods are not only applied to various benchmark datasets but also the trend following problem involving multi-sensor collected time series.

The remainder of this paper is organized as follows. Section II reviews some work related to FL and SVD-based model compression methods. The details of the proposed framework, including the SVD-based communication efficient method, and the multi-central model aggregation strategy, are described in Section III. Section IV presents the experimental results of the framework on benchmarks and the trend following tasks on coal mine electromagnetic radiation intensity. Finally, the conclusions are drawn in Section V.

II. Related Work

A. Existing FL Strategies

Recent research on FL mainly includes the reduction of communication costs, the optimization of aggregation strategies and the enhancement of privacy protection.

Regarding the reduction of communication costs, Zhu and Jin [16] proposed a multi-objective federated learning algorithm by taking overall communication costs and global test error as optimization targets. Then, a multi-objective optimization algorithm was used to search for the optimal structure for all the neural network in the clients. In order to further reduce the computational and communication costs incurred by evolutionary optimization, they [17] set master models on the server and introduced a double-sampling technique. Concretely, a randomly sampled sub-model of a master is transmitted to multiple randomly sampled clients for training without initialization. Based on [18], Mao et al. [19] proposed an adaptive quantization method that adaptively adjusts the quantization level according to the local gradient's update. Similarly, Yan et al. [20] presented a dynamic quantized stochastic gradient descent (SGD) method to explore the trade-off between communication costs and modelling error, experiments on different tasks demonstrate the effectiveness of their method.

As for the aggregation optimization of the central model, Yurochkin et al. [21] applied the Bayesian nonparametric mechanism to identify subsets of neurons in a local model that match neurons in other models. Then the matched neurons were uploaded to the server for forming the global model. Based on the different performance capabilities of various depth layers in neural network, Chen et al. [5] proposed an asynchronous update method for the client parameters to reduce the communication costs of each single round. Meanwhile, they presented a temporally weighted aggregation method by leveraging the previously uploaded parameters to improve the convergence speed meanwhile reducing the required communication rounds.

For the enhancement of privacy protection, a differential privacy method was adopted to decrease the influence of an individual's information when querying specific data repositories by adding Gaussian noise to the deep learning models [22]. Gayer et al. [23] introduced the differential privacy into federated learning, this technique does not show any information of the customers who have participated in the offline training, and further protect the entire private dataset from being attacked by other clients.

Besides that, some up-to-date techniques are gradually combined with FL. Jiang et al. [24] presented the FL as a natural source of practical applications for model agnostic meta-learning. According to their declaration, the objective of FL should be personalized performance, i.e., the shared global model only be a necessary partial step towards optimizing the personalized performance. To ease the domain shift among different nodes, Peng et al. [25] extended adversarial adaptation techniques to align the representations learned from different nodes with those of the target node. Additionally, a feature disentanglement method was developed to enhance the positive knowledge transfer.

B. SVD-Based Model Compression Approaches

Our framework aims to reduce communication costs by uploading the small-scaled singular values decomposed by SVD. In this subsection, we first review some common matrices decomposition methods, including triangular factorization [26], QR factorization [27], and SVD. Then, the SVD-based model compression methods are elaborated.

Triangular factorization factors a square matrix ($A_{m\times m}$) as the product of a lower triangular matrix ($L_{m\times m}$) and an upper triangular matrix ($T_{m\times m}$). The triangular factorization is formulated as:

$$A_{m\times m} = L_{m\times m} \cdot T_{m\times m}. \tag{1}$$

Triangular factorization plays a key role in solving square systems of linear equations and computing the determinant of a matrix [28]. As a comparison, QR factorization decomposes the matrix ($A_{m\times n}$) into a product of an orthogonal matrix ($Q_{m\times n}$), i.e., $QQ^T = I$, and an upper triangular matrix ($R_{n\times n}$). The formulation is shown as:

$$A_{m^\times n} = Q_{m\times n} \cdot R_{n\times n}. \tag{2}$$

QR decomposition is often adopted to solve the linear least squares and linear approximation problems [29]. SVD can obtain the full rank decomposition of arbitrary matrix $A_{m\times n}$ as the following form:

$$A_{m\times n} \;=\; U_{m\times m} \cdot \Sigma_{m\times n} \cdot V^T_{n\times n}, \tag{3}$$

where *U* and *V* are called left and right singular matrices, respectively. Σ is a diagonal matrix in which the diagonal elements are singular values. The singular values correspond to the important features implied in the matrix, and the importance degree is positively correlated with its values. In addition, the singular value matrix reflects the importance between the row of left singular vectors and the column of right singular vectors. Among these matrix factorization methods, only SVD can perform matrix compression by adjusting the proportion of singular values during the reverse process. The compression process can be reached as:

$$\begin{aligned} A_{m\times n} &= U_{m^\times m} \cdot \Sigma_{m^\times n} \cdot V^T_{n^\times n} \\ &\approx U_{m^\times r} \cdot \Sigma_{r^\times r} \cdot V^T_{r^\times n} \\ &= A'_{m^\times n} \end{aligned} \tag{4}$$

where r represents the preserving singular values while A'_{m^*n} denotes the compressed matrix with the same size as A_{m^*n}. Similar to SVD, the principal component analysis (PCA) [30] is equivalent to using only the right singular matrix in SVD i.e., the V^T, without considering the relationship between rows in the original matrix. To this end, SVD is adopted in this study to compress the model parameters for reducing communication costs.

SVD is usually adopted as a low-rank approximation to address large-scale parameters problems by representing the information within a matrix using the other matrix whose rank is smaller than the original matrix [31]. To obtain the proper rank configuration, Kim et al. [32] took the complete network as a whole and proposed an accuracy metric to measure the accuracy and complexity relationship. The calculation of overall accuracy was represented by the joint distribution of accuracy of each layer, and the joint distribution was calculated via product considering the independence. A metric that combined the PCA energy and the measurement-based approaches was developed to reduce the deficiency of individual metrics at low or high complexities. Similarly, Idelbayev [33] formulated the determination of the optimal rank of each layer as a mixed discrete-continuous optimization, which incorporates minimizing classification loss along with the model selection costs. They demonstrated the formulation could be optimized by interleaving SGD on the uncompressed net with SVD steps that can decide the optimal rank and model parameters. In [34], the researchers predefined the number of least significant singular values and conducted the SVD in an iterative "train-compress-retrain" method. The SVD was further combined with pruning and clustering compression. A greedy search algorithm was adopted to search the combination order of these approaches.

Zhou et al. [35] also adopted SVD in FL to reduce the communication costs, but we are completely different. In their work, all decomposed matrices, i.e., the *U*, *V*, and Σ were uploaded to the server, which increase the amount of transmitting data. On the server, the reverse-SVD was performed on all uploaded matrices, which increases the workload of server. In contrast, our method only transmits the singular value matrices of the dense layer to the server, then performs the reverse-SVD locally via the updated singular value matrices along with the local left/right singular matrix. As such, the critical information of the weights from the dense layer could be modified according to those from other clients using lower communication costs and local computation consumption. In our method, all matrices always keep full rank during the information update process. Therefore, the proposed framework does not belong to a low-rank approximation like

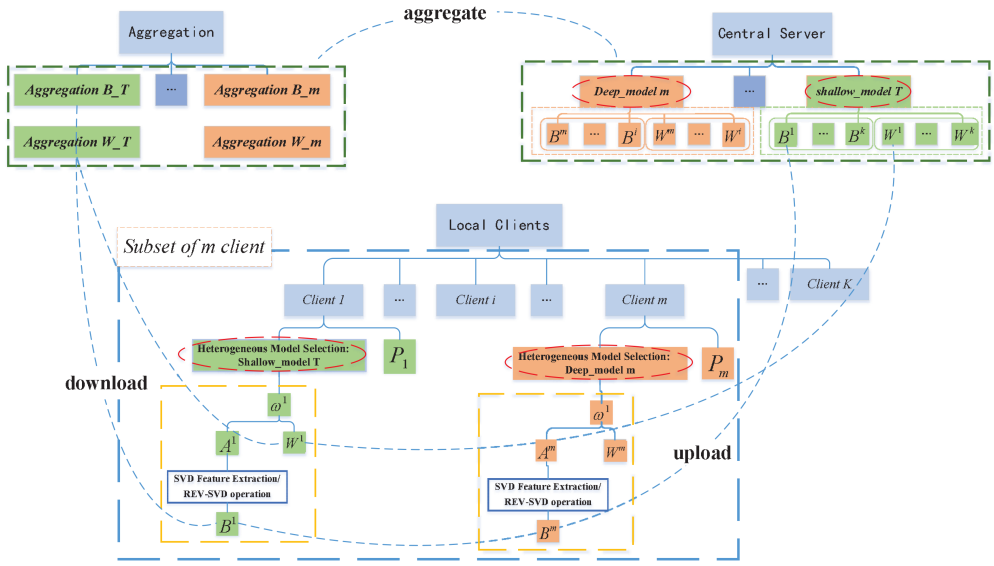


Fig. 1. The proposed heterogeneous structured federated learning framework with SVD-based low-communication method.

the [32], [34], but a full-rank communication efficient model sharing strategy.

III. Low-Communication Federated Learning With Heterogeneous Structured Models

A. Problem Formulation

FL has emerged as a promising field focused on training global machine learning models without implicitly accessing the training data stored on each client. In FL, the information uploaded by each participant to the server is no longer the original data, but the learned local model parameter (e.g., the weights). The currently prevalent paradigm usually formulates FL as an empirical risk minimization problem as follows:

$$\mathcal{L}_{fl}(\Theta;\mathcal{D}) = \sum_{i=1}^{n} \mathcal{L}_i(\Theta;\mathcal{D}_i). \tag{5}$$

where n is the number of clients participating in training, Θ represents the parameter sets of global model to be learned, and $\mathcal{L}_i(\Theta;\mathcal{D}_i)$ represents the loss of global model Θ on the local data with distribution $\mathcal{D}_i$ stored on client i. Let $\mathcal{L}_{con}$ represent the total learning loss for the conventional mode in which all datasets $\mathcal{D} = \mathcal{D}_1 \cup \ldots \cup \mathcal{D}_K$ are centralized to train a global model. Ideally, $\mathcal{L}_{fl}$ should be very close to $\mathcal{L}_{con}$. Given a non-negative number δ, the optimization of FL can be reached as

$$|\mathcal{L}_{fl} - \mathcal{L}_{con}| < \delta, \tag{6}$$

$$\Theta^* = \arg\min |\mathcal{L}_{fl} - \mathcal{L}_{con}|. \tag{7}$$

One of the defining characteristics of conventional FL is that all participating clients adopt the homogeneous local model while ignoring the difference between local samples in various clients, leading to low learning performance and high communication costs. To develop heterogeneous FL, the first problem we need to solve is the design of heterogeneous structured global models on the server. Subsequently, since there are different structured global models on the server, assigning proper global models to each client is another challenge we should consider. Furthermore, how to investigate effective local model training methods and efficient transmission strategies also are significant questions we have to settle. After the server receives the uploading contents from participating clients, the development of aggregation methods for heterogeneous models is the final problem to be solved. This study attempts to investigate the superiority of heterogeneous structured FL and thus deliver an example solution to the mentioned problems.

B. Framework

The proposed heterogeneous structured FL framework strengthened with high-efficient communication method is illustrated in Fig. 1. Like the canonical FL, our proposed framework contains the local model training and central parameter aggregation stages. However, there exist essential differences in each procedure. In the local model training, each client first selects its own local model according to the data size, as shown in the red circle module. In other words, the learning models adopted in different local clients could be with various structures. It is noted that this study aims to emphasize the superiority of adopting various structured central model rather than proposing a specific model selection criterion. The clients train their learning models based on the local dataset and upload the parameters to the central server. Instead of transmitting complete model parameters, the uploaded objects consist of the small-scale parameters from feature extraction layers (B^m) and the singular values matrices (W^m) of dense layers obtained by SVD. Subsequently, shown as in the green box, a multi-central aggregation mechanism is proposed for aggregating the heterogeneous structured local model parameters on the server. All clients download the aggregation from the central server. Furthermore, just as the yellow module, we perform the reverse-SVD on the aggregated

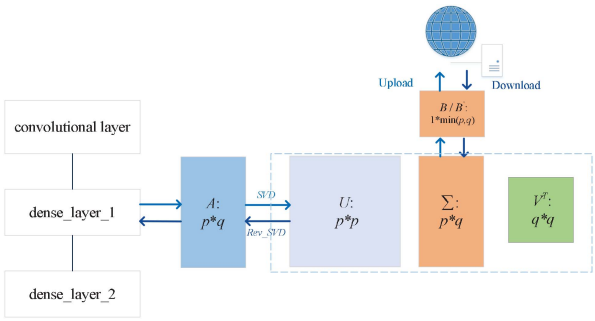


Fig. 2. Schematic diagram of SVD based feature extraction and updating.

singular value matrices (B_T) to obtain the updated parameters of dense layers. The aggregation (W_T) is directly taken as the updated parameters of the feature extraction module. Finally, the updated local model is regarded as the initialization for the following local training. These modules will be elaborated in the following subsections.

C. SVD-Based Communication Efficient FedAVG

In traditional FedAVG, all parameters of the models in participating clients are uploaded to the server for aggregation and update. The huge transmission amounts during the uploading and downloading require a high bandwidth and lead to high communication costs. Since the clients usually take neural networks as their local models, we analyze the characteristics of neural network and propose an efficient uploading/downloading strategy to reduce the size of transmission.

Neural networks, especially deep learning models, usually consist of two modules. The first one is feature extraction module containing many layers, such as the convolutional layers [36] or long short-term memory (LSTM) layers [37]. The other one is the task-specific (e.g., classification, regression, etc.) dense layers. The feature extraction layers are mainly used to extract high-order features, which plays an essential role in ensuring the performance of entire model. The dense layers aim to non-linearly combine the extracted features and realize the final tasks. To reduce the communication costs without causing much accuracy loss, we thus utilize SVD to decompose the parameters from dense layers of the client models, i.e., the high-dimensional parameters. Then, instead of uploading complete weight matrices in the canonical FedAVG, only the singular value matrix is uploaded to participate in the aggregations. The parameters of feature extraction module are still fully uploaded to ensure the generalization ability of feature extraction modules.

Taking CNN as an example, the entire process of utilizing SVD with the dense layer in the communication is shown in Fig. 2. We first use SVD to decompose the weight matrix $A_{p\times q}$ of Dense_1 layer, i.e., the first dense layer followed the convolutional layer. For $A_{p\times q}$, p and q are the numbers of the units in the input and hidden layers respectively. The p is also the number of the feature extraction output units or the dimension of the extracted features. After getting the singular value matrix $\sum_{p\times q}$ along with the left/right singular matrix, each client uploads the diagonal elements of singular value matrix B to the server for aggregation. Since the singular values correspond to the important features implied in the matrix, the local importance degree can be updated by aggregating information from other clients. With the aggregated feature matrix B', the reverse SVD operation is performed on the stored left/right singular matrix to update the weights of dense layer. As the parameters of feature extraction module update in a completely uploading mode to ensure the generalization, the stored left/right singular matrices are expected to maintain the personalization of each local model.

The total communication costs of the dense weights can be greatly reduced with SVD. For the i-th client model, the weight matrices of layers which are not processed with SVD-based key feature extraction method are denoted as W_i, while the weights of dense layers are represented as $A_{p_i\times q_i}$ and B_i before and after performing SVD. The amount of uploaded/downloaded data decreases from $p_i \times q_i$ to $\min(p_i, q_i)$ after SVD, which is merely $1/q_i$ or $1/p_i$ of the original communication costs. From the perspective of all clients involved in FL, we assume the number of participating clients is l, the amount of uploaded/downloaded data in canonical FL is $2\sum_{i=1}^{l}(|W_i| + |A_{p_i\times q_i}|)$, and that reduces to $2\sum_{i=1}^{l}(|W_i| + |B_i|)$ after performing SVD on the parameters of the dense layer, the differences between them can be calculated as:

$$2\sum_{i=1}^{l}(|A_{p_i\times q_i}| - |B_i|) = 2\sum_{i=1}^{l}(\min(p_i, q_i) \times (\max(p_i, q_i) - 1)). \quad (8)$$

The more participating clients, the more effective the proposed feature uploading algorithm is. Furthermore, the algorithm only transmits key features and keeps the left/right singular matrix in the client, which further reduces the possibility of privacy leakage and enhances the confidentiality of the system.

D. Aggregation Strategy for FL With Heterogeneous Structured Client Models

As shown in Fig. 1, the proposed FL framework is quite different from the canonical one, the traditional aggregation method for homogeneous models is thus no longer applicable. We present a multi-central model aggregation mechanism for heterogeneous structured models in this section.

For the i-th client model, the parameters to be aggregated include two parts, i.e., W_i and B_i, which represent different contents of the model and should be separately aggregated accordingly. First, clients are clustered according to the structure of their adopted model, and those clients with the same models are clustered into same class. Then, we further divide the communication rounds into specific and regular rounds. In the former ones, intact parameters of task-specific layers are uploaded, which is expected to obtain a shared central model and avoid local optimal. In the latter kind of communication rounds, the singular value matrices are uploaded to tackle the communication bottleneck. On the server, the uploaded W_i and B_i of the clustered models are separately aggregated according to any aggregation optimization, e.g., the average weighted one.

Algorithm 1: Server Component of SVD-HCMFL.

1: **function** ServerExecution ▷Run on the server
2: Initialize $\omega_{deep,1}$for deep_model
3: Initialize $\omega_{shal,1}$for shallow_model
4: Initialize set_value, svd_index and agg_round
5: **for** each round $t \in 1, 2, \ldots, T$ **do**
6: $m \leftarrow \max(C \cdot K, 1)$
7: $S_t \leftarrow$ random set of m clients
8: **for** each client $k \in S_t$ in parallel **do**
9: $B'^k_{t+1}, W'^k_{t+1}, flag \leftarrow ClientUpdate$ $(t, k, \omega_{deep,t}, \omega_{shal,t}, set\_value, svd\_index,$ $agg\_round)$
10: **if** $flag$==$deep$ **then**
11: $W'^k_{deep,t+1}, B'^k_{deep,t+1} \leftarrow W'^k_{t+1}, B'^k_{t+1}$
12: **else**
13: $W'^k_{shal,t+1}, B'^k_{shal,t+1} \leftarrow W'^k_{t+1}, B'^k_{t+1}$
14: **end if**
15: **end for**
16: $\omega_{deep,t+1} \leftarrow \sum_{k=1}^{m_d} \frac{n_k}{n} \left(W'^k_{deep,t+1} \cup B'^k_{deep,t+1}\right)$
17: $\omega_{shal,t+1} \leftarrow \sum_{k=1}^{m_s} \frac{n_k}{n} \left(W'^k_{shal,t+1} \cup B'^k_{shal,t+1}\right)$
18: **end for**
19: **return** $\omega_{deep,T+1}$, $\omega_{shal,T+1}$
20: **end function**

Clearly, the number of the aggregated model is determined by the types of learning models, which is different from the traditional one that only uses one aggregation model. In addition, the aggregation mechanism can vary for different kinds of local models. After aggregation, the obtained weights and singular values are downloaded to the clients for update.

It makes sense to cluster users according to the size of the dataset generated by users from the perspective of personalized recommendation. Clients with a small dataset can be considered users in the early stage of cognition, and they generally have less knowledge of the items and fluctuant preferences. These clients are gathered to expect to broadly expand their knowledge of the unknown through exchanging fluctuant cognition to build new preferences gradually. As a comparison, the clients containing large amounts of datasets correspond to users with stable cognition and personal preference. They rarely need to develop new preferences through fluctuant cognition from others during the information exchange. What's more, the cognition obtained from clients who are in the early stage of cognition may even become a distraction.

The pseudo-code for the two main components of the proposed Heterogeneous structured Client Model Federated Learning with SVD-based communication efficient method (SVD-HCMFL), i.e., the implementation on the server and local clients, are given in Algorithms 1 and 2, respectively.

The specific server component is further explained in the following contents :

1) *Initialization:* Lines 2-4 initialize parameters ω_{deep} and ω_{shal} for the deep_model and shallow_model, respectively, as well as the set_value, svd_index and agg_round.

2) *Communication round:* The whole process is divided into several loops. In each loop, Line 6 specifies *m*, i.e., the number of participating clients, according to the number of local clients K and the fraction of participating clients C per round. The participating subset S_t are randomly selected in Line 7. In lines 8-15, sub-function $ClientUpdate$ is called in parallel to update the uploaded parameters from deep/shallow models, and then get the $W'^k_{deep,t+1}$, $B'^k_{deep,t+1}$ and $W'^k_{shal,t+1}$ as well as $B'^k_{shal,t+1}$ according to the values of $flag$. In Lines 16 and 17, based on the dimensions of parameters from selected model, the parameters of feature extraction layers and the transmitted contents of dense layers, i.e., the original parameters matrices in specific rounds or singular values in regular rounds, from deep/shallow networks are respectively aggregated according to the following formula. Therefore, the new parameter sets $\omega_{deep,t+1}$ and $\omega_{shal,t+1}$ are obtained:

$$\omega_{deep,t+1} = \sum_{k=1}^{m_d} \frac{n_k}{n} \left(W'^k_{deep,t+1} \cup B'^k_{deep,t+1}\right), \tag{9}$$

$$\omega_{shal,t+1} = \sum_{k=1}^{m_s} \frac{n_k}{n} \left(W'^k_{shal,t+1} \cup B'^k_{shal,t+1}\right), \tag{10}$$

where n_k is the sample size of the k-th client, n is the total number of training samples. m_d and m_s represent the number of clients that select deep model and shallow model, respectively.

As for the implementation of the local model update (Algorithm 2), the input variables include t, k, ω_{deep}, ω_{shal}, set_value, svd_index and agg_round, where t is the number of communication rounds, k indicates the index of selected client. The set_value is used to determine the $flag$ value and further judge which kind of model is being updated. For instance, $flag$==$deep$ means the function is updating the parameters of deep_model. The ω_{deep} and ω_{shal} represent parameter sets of deep and shallow model, respectively. svd_index represents the execution index for extracting key features of the parameters from which dense layer. Finally, agg_round determines the type of communication round, i.e., the the uploaded item is the singular value matrices obtained by SVD or the intact parameter matrices. Line 2 splits data into batches, concretely, P_k corresponds to the set of indices of local dataset on client k, with $n_k = |\mathrm{P}_k|$. Lines 3-7 first determine the value of $flag$ that can only be $deep$ or $shal$, and then transfer the model parameters to a temporary matrix ω_{temp}. Lines 8-15 mean that each client directly takes the initial parameters from the server as their local parameter when $t = 1$ or at the specific communication round. In other cases, client has to firstly perform reverse-SVD decomposition to get the updated B^{temp}. Then, the B^{temp} and W^{temp} constitute a new updated parameter set ω that are taken as the initial parameters for further local training. The $\overline{\langle\cdot\rangle}$ represents all locations except current index. Lines 16-20 execute local SGD to train local models, where B and E are the local mini-batch size and the number of local training epochs, respectively. η denotes the learning rate. The uploaded items are determined based on the number of agg_round in lines 21 to 26. The major difference between our aggregation methods and the conventional FedAVG lies in that our method individually

Algorithm 2: Client Component of SVD-HCMFL.

function ClientUpdate $t, k, \omega_{deep}, \omega_{shal}, set\_value, svd\_index, agg\_round$ ▷Run on client k
  B $\leftarrow$ (splite P_k into batches of size B)
  if $\mathrm{P}_k > set\_value$**then**
    $flag \leftarrow deep$ and $\omega_{temp} \leftarrow \omega_{deep}$
  else
    $flag \leftarrow shal$ and $\omega_{temp} \leftarrow \omega_{shal}$
  end if
  if $t == 1$ or $t \,\%\, agg\_round == 0$ **then**
    $\omega \leftarrow \omega_{temp}$
  else
    $B^{temp} \leftarrow rev\_svd(\omega_{temp}[-1])$
    $W^{temp} \leftarrow \omega_{temp}[:-1]$
    $\omega[svd\_index] \leftarrow B^{temp}$
    $\omega[\overline{\langle svd\_index \rangle}] \leftarrow W^{temp}$
  end if
  foreach local epoch epo from 1 to E **do**
    for $b \in$ B **do**
      $\omega \leftarrow \omega - \eta \nabla \ell(\omega; b)$
    end for
  end for
  if $t \,\%\, agg\_round == 0$ **then**
    $B \leftarrow \omega[svd\_index]$
  else
    $B \leftarrow svd(\omega[svd\_index])$
  end if
  $W \leftarrow \omega[\overline{\langle svd\_index \rangle}]$
  return $B, W, flag$ to server
end function

aggregates the parameters of each module from heterogeneous structured networks. Besides that, the deep and shallow networks are independently aggregated, so that their aggregation processes can be taken as two noninteracting processes. To this end, we can verify the effectiveness of our aggregation method following the same procedure of proving the convergence of FedAVG [38].

IV. Experimental Results and Analyses

A. Datasets and Parameter Settings

The proposed framework is first verified by applying it to three benchmark classification tasks, i.e., the MNIST, CIFAR-10, CIFAR-100, and Human Activity Recognition (HAR) dataset [39]. Four different FL algorithms, i.e., FedAVG [5], federated low-rank algorithm (FedLR) [35], FedLR-HCMFL, and the temporally weighted aggregation asynchronous federated learning (ASTW_FedAVG) [6] are taken as comparing methods. Their performance is evaluated in terms of classification accuracy, communication cost, and time consumption. FedLR adopts SVD to calculate the low-rank approximation of convolutional layers and uploads the decomposed matrices. In order to find optimal compression ratio, FedLR minimizes the differences of the first-order increment between the loss

TABLE I
Parameter Settings for Heterogeneous Structured CNNs

Type	Layer (Activation Function)	Parameter Size
DCNN	conv2d_1(relu)	[3×3×1(3)+1]×32
	conv2d_2(relu)	[3×3×32+1]×32
	conv2d_3(relu)	[3×3×32+1]×64
	conv2d_4(relu)	[3×3×64+1]×64
	dense_1(relu)	[1024(1600)+1]×512
	dense_2(softmax)	[512+1]×10(100)
SCNN	conv2d_1(relu)	[3×3×1(3)+1]×32
	conv2d_2(relu)	[3×3×32+1]×64
	dense_1(relu)	[1600(2304)+1]×512
	dense_2(softmax)	[512+1]×10(100)

rate and compression ratio of the parameter matrix. To further validate the superiority of our framework over FedLR under heterogeneous FL, we let FedLR work in synergy with heterogeneous structured client models federated learning (HCMFL) and term it as FedLR-HCMFL. The introductions of FedAVG and ASTW_FedAVG can be found in Section I. The superiority of our framework is demonstrated from the following four aspects: (1) feasibility of data size-based heterogeneous structured neural network selection; (2) performance of HCMFL framework with the averaging-based aggregation method; (3) performance of the FedAVG algorithm articulated with SVD (SVD-FedAVG); (4) effectiveness of the proposed SVD-HCMFL framework.

The MNIST dataset has ten different kinds of digits and every digit is a 28×28-pixel gray-scaled image.The whole dataset contains 60,000 training images and 10,000 test images. CIFAR-10 dataset has ten different kinds of items, and consists of 50,000 training images and 10,000 test images, each one corresponds to a 32×32-pixel colored image. The images from CIFAR-100 have the size same as those from CIFAR-10, but they are categorised into 100 classes. For these three datasets, two types of heterogeneous structured networks, i.e., the deep CNN (DCNN) and shallow CNN (SCNN) are adopted. The specific structural parameters are shown in Table I. Among the parameters, conv2d_*n* and dense_*n* represent the *n*-th convolutional layer and dense layer in the network, respectively. Parameter size indicates the number of parameters of that layer, where the values in parentheses correspond to the parameters when processing the CIFAR-10 dataset. Taking the parameters in conv2d_1 as example, 3×3 represents the 3×3 convolutional kernel, 1(3) corresponds to the number of channels of the pictures from MNIST and CIFAR-10 datasets, 1 indicates the number of basis, and 64 means the number of convolutional kernels.

In the HAR dataset, each single data corresponds to a sequence of images with a label. Thus the LSTM network, which carries an inductive bias for sequential data, is adopted as the classifier. We mainly consider 6 different kinds of human activities in this research. The adopted LSTM consists of two LSTM layers and two dense layers, the specific parameters are shown in Table II.

To simulate the environment of federated learning, we divide the above datasets into five and ten unbalanced subsets using the method given in [6]. Each subset is treated as a single client. The distributions of data amounts in different clients are

TABLE II
PARAMETER SETTINGS FOR LSTM

Layer (Activation Function)	Parameter Size
lstm_1	[9+20+1]×80
lstm_2	[20+10+1]×40
dense_1(relu)	[10+1]×128
dense_2(softmax)	[128+1]×6

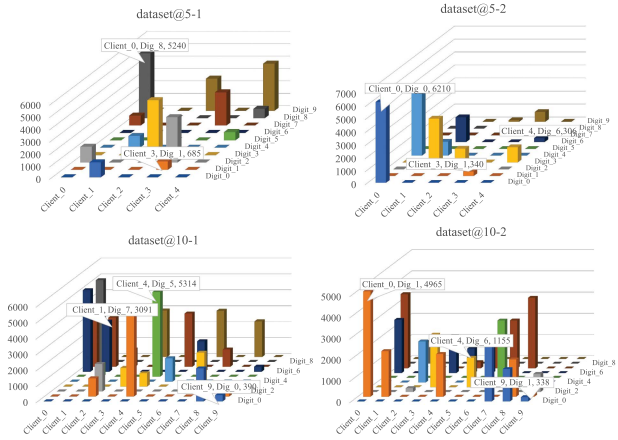


Fig. 3. Part of 3-D column charts of pre-generated datasets.

extremely unbalanced, specifically, we assign less than 2,500 data samples to most clients but around 10,000 data points to the rest in five clients setting. We further expand the gaps and only provide large portions of clients with about a hundred data for ten clients. In order to validate the generalization of our proposed framework, we set various client numbers and design different non-i.i.d data distributions for each client. The divided subset is represented as $dataset@client\_num$, where $dataset$ is the name of the original dataset, i.e., MNIST, CIFAR-10, CIFAR-100, and HAR. $client$ denotes the total number of the divided subsets, and the index of each client is denoted as num. Parts of subsets distribution for the MNIST with five or ten clients are presented in Fig. 3, where the height and color of each histogram respectively represent the size and categories of the samples in different client. The "digit" shown on the y-axis means the labels of corresponding handwriting digits. We assign identical distribution and data scale for the divided subset with the same index for the CIFAR-10 and CIFAR-100 datasets. Under our simulation setting, the generated data points scatted in different clients are typically unbalanced and follow the non-i.i.d.

The evaluation of central model is conducted with the following equation:

$$Acc(\omega_t) = \frac{n_k}{n} \sum_{k=1}^{K} acc_k(\omega_t, P_k), \tag{11}$$

where the $Acc(\omega_t)$ is the accuracy of central model with the aggregated parameters ω_t, while the $acc_k(\omega_t, P_k)$ indicates the accuracy obtained by testing on the local dataset P_k, t represents the current communication round, K means the number of participating clients. In this work, the experimental accuracy is

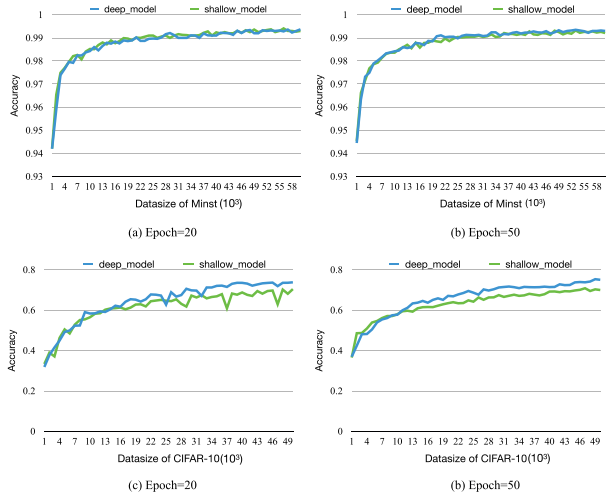


Fig. 4. The accuracy of DCNN and SCNN w.r.t the size of training set under i.i.d setting.

obtained by calculating the average results of five repeated experiments with different random seeds on each divided dataset.

B. Experimental Results and Analysis

1) Effectiveness of Heterogeneous Structured Models: In this section, we validate the superiority of heterogeneous structured models over the models with the same structure. First, we compare the test accuracy of DCNNs and SCNNs trained on i.i.d datasets with different amounts of training samples. Specifically, the training sets of MNIST and CIFAR-10 datasets are segmented and expressed as $T_v^{mni} = Data_v[0 : v \times 1000], v = 1, \ldots, 60$ and $T_v^{cif} = Data_v[0 : v \times 1000], v = 1, \ldots, 50$. The test accuracies w.r.t. the size of training sets are demonstrated in Fig. 4. Two observations can be made: (1) For CIFAR-10, when the training set size is relative small (less than about 8000), the test accuracy of SCNN is higher than that of DCNN. As the size increases, the accuracy of DCNN gradually becomes higher than SCNN. Although the MNIST dataset is relatively simple, similar results can also be observed. (2) In different training epochs, i.e., 20 and 50, the variations of the classification accuracy of CNNs on each individual dataset are similar to the above observation, indicating that the proposed heterogeneous structured model selection strategy based on the data size is reasonable.

Subsequently, we conduct experiments on the divided non-i.i.d subset datasets. Clients with different amounts of datasets select varied models based on the obtained threshold, and this case is called heterogeneous mode. In contrast, we take the case that all clients adopt models with identical structure as homogeneous mode. The selected models are trained on local samples within each single client, but they do not communicate with central server. The average accuracy of each divided subset is presented on Table III . If the results obtained by both DCNN and SCNN in heterogeneous mode are higher than those of homogeneous mode, the results are bolded. From the results, we can find the average accuracies achieved by heterogeneous

TABLE III
AVERAGE ACCURACY OF DCNN AND SCNN IN DIFFERENT MODES ON NON-I.I.D DIVIDED DATASETS

Dataset@client_num	Heterogeneous Mode		Homogeneous Mode	
	DCNN	*SCNN*	*DCNN*	textitSCNN
MNIST@5-1	**94.32**	**93.50**	92.75	91.65
MNIST@5-2	98.00	94.52	94.93	96.41
MNIST@5-3	97.26	95.51	96.87	95.86
MNIST@5-4	**97.06**	**94.76**	94.26	94.70
MNIST@5-5	**97.11**	**95.82**	95.94	94.99
MNIST@10-1	**96.30**	**95.94**	94.37	93.80
MNIST@10-2	98.48	94.27	96.19	95.63
MNIST@10-3	**97.43**	**92.45**	93.04	90.20
MNIST@10-4	**97.98**	**92.63**	92.45	90.97
MNIST@10-5	97.64	90.41	91.25	90.58
Average_accuracy	**97.16**	**93.98**	94.21	93.48
CIFAR-10@5-1	76.70	71.66	72.09	72.67
CIFAR-10@5-2	65.95	73.97	66.47	68.87
CIFAR-10@5-3	**71.94**	**76.12**	71.00	74.17
CIFAR-10@5-4	**69.97**	**65.89**	66.62	65.81
CIFAR-10@5-5	**82.61**	**72.78**	79.29	71.13
CIFAR-10@10-1	**66.87**	**70.99**	68.06	67.82
CIFAR-10@10-2	77.23	72.48	68.51	74.70
CIFAR-10@10-3	**66.47**	**73.53**	65.09	69.39
CIFAR-10@10-4	69.57	72.84	70.31	71.61
CIFAR-10@10-5	**70.17**	**72.88**	68.98	71.51
Average_accuracy	**71.75**	**72.31**	69.64	70.77

TABLE IV
HYPERPARAMETERS FOR HCMFL

Parameters	Value
K	{10, 5}
m	{8, 4}
agg_round	5
set_value	8000
optimizer	Adam
batch size	50
local epoch	20

structured models are higher than those of homogeneous models, which demonstrates the superiority of adopting models with heterogeneous structures under varied amounts of training samples.

2) Performance of HCMFL: The effectiveness of HCMFL on improving convergence accuracy and reducing communication costs are demonstrated by comparing with the FedAVG and FedLR. We record the convergence accuracy when the accuracy fluctuation is less than 4% within 300 communication rounds. Otherwise, the algorithm is thought to fail in convergence. The communication costs are measured by the consumed communication rounds multiplying the transmitted amount of each round. The costs of SCNN using the FedAvg are selected as the baseline. Specific hyperparameters are shown in Table IV-A. Corresponding experimental results can be found in Table V, where the first column, i.e., the "*Dataset@client-num*," represents various divided datasets. For the statistics of HCMFL, "DCNN#" and "SCNN#" represent the number of adopted DCNN and SCNN in the algorithm, respectively. The "*Accuracy [Round]*" records the convergence accuracy of both DCNNs and SCNNs as well as their required communication rounds. It is noted that these results are counted only when both heterogeneous structured models converge. As a contrast, the "*Accuracy [Round]*" in FedLR and FedAVG denote these metrics of individual model. *C.Cost(T.Cost)* represents the required communication costs and time costs. Best results are bolded.

From Table V, we can conclude that: (1) Generally, the proposed HCMFL can achieve higher accuracy, especially for CIFAR-10 dataset. Taking the 10-4 dataset as an example, after 130 communication rounds, the average convergence accuracy of DCNN is 70.7% and that of SCNN is 78.8%. By contrast, after 136 rounds, the accuracy of DCNN in FedAVG is 61.4% while the SCNN fails to converge. The reason could be that clients select proper model to reduce the possible overfitting and underfitting, thereby improving the accuracy of the aggregated central model. FedAVG reaches the second-best results as the homogeneous structured models may struggle to deal with the local samples with significantly different sizes. FedLR also adopts homogeneous structured models but achieves the lowest accuracy. This is mainly because FedLR individually determines the compression ratio of each individual element of the parameter matrices while ignoring the connection between these elements, which may cause the compressed matrices fail to cover all significant information. It thus inevitably hinders the convergence accuracy. (2) When the algorithm achieves convergence, however, the performance of HCMFL framework on reducing communication costs of is not remarkable. This is mainly because the DCNNs and SCNNs have to upload all parameter matrices, resulting in high costs in each communication round. FedLR, while reducing the amount of content to transmit, does not significantly reduce communication costs because it increases the number of communication rounds required for convergence. (3) In HCMFL, when the number of participating clients is small, the convergence is not satisfying. Taking the CIFAR-10 dataset as an example, when the number of divided clients is 5, the algorithm fails to converge twice. The reason could be that if the total number of clients is small, the quantity of deep/shallow models that can participate in aggregation is too small. It is thus difficult to extract the common features of different clients.

3) Feasibility of SVD-FedAVG: The effectiveness of our proposed SVD-FedAVG is validated by comparing with FedAVG in terms of communication efficiency along with the convergence accuracy. The CIFAR-10 and HAR datasets are selected since these two tasks are relative complex. The results of 10 different segments on CIFAR-10 and HAR dataset are given in Figs. 5 and 6. In these figures, the red and blue vertical dashed lines indicate the communication rounds required for SVD-FedAVG and FedAVG when achieving the same accuracy, respectively. Accordingly, we can find that (1) For the same accuracy, the required communication rounds of SVD-FedAVG is less than that of the FedAVG in most cases. As SVD-FedAVG transmits much smaller amount of data than FedAVG does in each round, we can conclude that the SVD-FedAVG can effectively reduce the communication costs. (2) Although the accuracy of central models obtained by SVD-FedAVG is more volatile than the FedAVG, it is generally higher in both the general accuracy and highest accuracy. According to the characteristics of SVD, the value of singular values is positively correlated with the importance degree of features implied in the matrix, and

TABLE V
CONVERGENCE ACCURACY AND CORRESPONDING RESOURCE COST OF HCMFL

Dataset@client_num	HCMFL				FedLR		FedAVG	
	DCNN#	SCNN#	*Accuracy[Round]*	*C.Cost(T.Cost)*	*Accuracy[Round]*	*C.Cost(T.Cost)*	*Accuracy[Round]*	*C.Cost(T.Cost)*
MNIST@5-1	2	3	–	–(–)	93.6[166]/94.5[196]	1.15(6306)/1.97(**5074**)	94.9[151]/**95.9**[**101**]	1.05(3956)/**1.00**(6733)
MNIST@5-2	3	2	96.8/95.1[**136**]	0.61(6005)	94.3[166]/95.4[206]	0.59(6926)/1.07(7018)	94.9[**136**]/96.7[196]	**0.49**(**4313**)/1.00(6934)
MNIST@5-3	2	3	**98.8/95.6**[**81**]	**1.00**(**5341**)	–/–	–(–)/–(–)	–/–	–(–)/–(–)
MNIST@5-4	1	4	94.1/93.7[226]	2.27(9532)	96.5[196]/96.8[241]	1.81(9107)/3.22(7778)	**98.7**[226]/97.3[**76**]	2.20(6958)/1.00(5661)
MNIST@5-5	2	3	**95.1/94.3**[**121**]	0.66(10234)	92.9[196]/89.1[256]	0.63(9799)/1.18(8991)	93.5[181]/91.7[220]	**0.58**(**5920**)/1.00(7727)
MNIST@10-1	3	7	**99.4/97.6**[151]	1.32(8211)	–/–	–(–)/–(–)	–/96.9[**91**]	–(–)/1.00(**5287**)
MNIST@10-2	2	8	**97.1/96.7**[111]	1.54(7765)	94.8[166]/95.8[196]	2.12(14838)/3.62(12232)	95.8[208]/96.7[**55**]	2.67(16979)/1.00(**3592**)
MNIST@10-3	3	7	**97.3/96.2**[116]	0.44(7582)	94.7[106]/94.8[151]	0.35(7769)/0.73(7665)	95.8[**61**]/95.2[211]	**0.20**(**4419**)/1.00(10311)
MNIST@10-4	3	7	95.6/96.7[**111**]	**0.42**(**6774**)	94.1[181]/95.7[241]	0.6(12062)/1.16(11426)	96.5[196]/97.7[211]	0.66(13691)/1.00(10025)
MNIST@10-5	4	6	98.5/96.4[**91**]	0.55(6368)	96.6[136]/94.9[166]	0.7(8935)/1.24(8338)	97.2[61]/96.1[136]	**0.32**(**4161**)/1.00(6243)
CIFAR-10@5-1	2	3	–/–	–(–)	68.7[191]/65.7[206]	0.75(7456)/1.11(10023)	**70.3**[**186**]/68.8[**186**]	**0.74**(**6705**)/1.00(8087)
CIFAR-10@5-2	3	2	78.1/74.3[220]	1.15(17312)	69.6[171]/67.67[251]	0.77(8917)/1.56(9257)	74.5[**102**]/72.6[161]	**0.47**(**4956**)/1.00(7885)
CIFAR-10@5-3	2	3	70.4/69.1[130]	1.02(7523)	–/68.3[121]	–(–)/1.04(7325)	–/**70.5**[**116**]	–(–)/**1.00**(**6396**)
CIFAR-10@5-4	1	4	**72.2/75.7**[200]	1.18(11334)	66.4[191]/63.7[196]	0.87(10051)/1.21(10618)	69.1[**151**]/62.5[161]	**0.69**(**7199**)/1.00(8914)
CIFAR-10@5-5	2	3	–/–	–(–)	–/–	–(–)/–(–)	–/–	–(–)/–(–)
CIFAR-10@10-1	3	7	70.5/72.8[**80**]	**0.38**(**6191**)	67.8[271]/67.9[271]	1.01(13571)/1.38(9932)	71.6[196]/70.8[196]	0.74(10005)/1.00(7430)
CIFAR-10@10-2	2	8	70.6/72.2[144]	0.78(9836)	66.7[166]/72.1[181]	0.69(9510)/1.03(5528)	67.4[**121**]/73.17[176]	**0.51**(7026)/1/00(**5141**)
CIFAR-10@10-3	3	7	71.8/65.9[**96**]	1.25(8115)	62.4[226]/60.5[161]	0.72(10462)/0.7(10040)	66.2[**96**]/63.11[230]	**0.31**(**5017**)/1.00(7950)
CIFAR-10@10-4	3	7	**70.7/78.8**[**130**]	1.19(8841)	59.7[271]/–	1.97(11827)/–(–)	61.4[136]/–	**1.00**(**6308**)/–(–)
CIFAR-10@10-5	4	6	75.5/68.4[64]	**0.51**(**4063**)	73.3[171]/67.2[226]	1.11(14686)/2.01(10223)	**77.4**[136]/69.2[**112**]	0.90(10923)/1.00(5025)
CIFAR-100@5-1	2	3	44.7/51.3[211]	**0.69**(5183)	43.9[231]/45.7[256]	0.66(5442)/0.97(5037)	46.9[261]/48.6[246]	0.71(5264)/1.00(**4958**)
CIFAR-100@5-2	2	3	**47.8/45.5**[246]	0.89(5871)	42.2[**216**]/42.5[261]	**0.68**(5407)/1.09(5359)	44.2[236]/43.6[261]	0.83(**4714**)/1.00(5244)
CIFAR-100@5-3	3	2	36.0/54.3[246]	0.94(6862)	45.8[201]/43.5[271]	**0.64**(**5471**)/1.13(5903)	47.7[236]/44.8[246]	0.78(5837)/1.00(5637)
CIFAR-100@10-1	4	6	41.6/36.8[**201**]	**0.71**(**7529**)	39.6[251]/41.2[281]	0.78(10472)/1.15(8734)	41.8[241]/43.4[276]	0.86(8238)/1.00(8240)
CIFAR-100@10-2	3	7	35.1/37.8[256]	0.82(8291)	37.9[**221**]/35.1[261]	**0.64**(7357)/1.01(**6764**)	**39.6**[256]/36.6[256]	0.75(7105)/1.00(6450)
CIFAR-100@10-3	5	5	43.6/33.6[**226**]	0.73(9651)	44.2[216]/40.2[256]	**0.59**(10697)/0.93(**9563**)	**45.5**[271]/43.9[266]	0.74(10476)/1.00(9691)
CIFAR-100@20-1	11	9	28.8/32.0[**246**]	0.85(42179)	26.3[256]/30.3[**246**]	**0.75**(43287)/0.95(40808)	28.9[256]/32.2[276]	0.81(42398)/1.00(**40614**)
CIFAR-100@20-2	10	10	**28.4/28.8**[251]	0.84(40956)	25.7[271]/25.6[256]	0.77(41423)/0.97(38645)	25.4[261]/27.8[**241**]	**0.69**(40948)/1.00(**37844**)

the sum of the first 10% of singular values almost occupies all the sum. Therefore, through aggregating singular value matrices, the noisy information implied in the weights of dense layers can be weakened, which makes the updated models concentrate more on their own tasks. That's the reason why the aggregated central model may achieve higher accuracy.

In order to investigate the effectiveness of the updated singular value matrix on the subsequent training, we also visualize the training accuracy w.r.t the communication rounds on CIFAR-100 datasets in Fig. 7. Under identical communication round, the training accuracy of SVD-FedAVG is generally higher than that of FedAVG. Furthermore, the SVD-FedAVG achieves more stable training accuracy than the FedAVG does. The promotion could be attributed to that the SVD-FedAVG effectively maintains the personalization of each client by only updating the singular values of dense layer. Since the singular values correspond to the degree of importance of features implied in matrix, the key features of dense layers can be enhanced via aggregating information from other clients.

4) Effectiveness of SVD-HCMFL: In this section, the performance of SVD-HCMFL is further investigated. The settings of HCMFL are the same as those given in Section IV-B2, and the SVD operation mode is consistent with that in Section IV-B3. In order to sufficiently demonstrate the superiority of the proposed SVD-HCMFL, we compare it with the FedLR-HCMFL, SVD-FedAVG, and ASTW_FedAVG on the MNIST, CIFAR-10, and CIFAR-100 datasets. The results are shown in Table VI whose implications are the same as those in Table V. In addition, the comparison of communication costs is based on the baseline of Table V.

According to the results in Table VI, the following observations can be obtained: (1) Compared with the heterogeneous FedLR-HCMFL, the SVD-HCMFL achieves better performance on most divided datasets no matter in accuracy, communication costs, and time costs. Specifically, the DCNN and SCNN built in SVD-HCMFL framework can achieve averaged 6.82% and 6.35% promotion over three datasets. In addition, the computation costs and time consumption of SVD-HCMFL only occupy 20.14% and 69.37% of FedLR-HCMFL, respectively. FedLR-HCMFL struggles to achieve comparable accuracy since it fails to capture the connection between each element within high-dimensional parameter matrices, which inevitably hinders the convergence accuracy and speed while increasing the computation and transmission costs. As a comparison, in SVD-HCMFL, the parameters of feature extraction module are fully uploaded to ensure the generalization ability of feature extraction modules. Meanwhile, we upload the extracted singular values of fully connected layers instead of whole parameter matrices to reduce communication costs. (2) Compared with the homogeneous FL frameworks, SVD-HCMFL also achieves decent performance on both improving the convergence accuracy and reducing communication costs. In contrast with SVD-FedAVG and ASTW_FedAVG, we can observe the averaged accuracy increase of DCNNs and SCNNs within SVD-HCMFL reach a ratio of 1.84% and 3.94% over three datasets, respectively. The improvements could be attributed the adoption of heterogeneous structured model and the key feature enhancement of dense layers in SVD-FedAVG. As for the convergence, SVD-HCMFL method can facilitate most models to converge and greatly reduce the communication costs. Compared with ASTW_FedAVG, in the MNIST and CIFAR-10 classification tasks, the averaged ratio of communication cost required for the model to converge only occupies 37.49% and 37.41%. In the most complicated CIFAR-100, the proposed framework is able to averagely save 64.25% of communication resources. Between SVD-HCMFL and SVD-FedAVG, the communication costs of DCNNs in SVD-FedAVG should be the

TABLE VI
CONVERGENCE ACCURACY AND CORRESPONDING RESOURCE COSTS OF SVD-HCMFL FRAMEWORK

Dataset@client_num	SVD-HCMFL		FedLR-HCMFL		SVD-FedAVG		ASTW_FedAVG	
	Accuracy[Round]	*C.Cost(T.Cost)*	*Accuracy[Round]*	*C.Cost(T.Cost)*	*Accuracy[Round]*	*C.Cost(T.Cost)*	*Accuracy[Round]*	*C.Cost(T.Cost)*
MNIST@5-1	–	–(–)	–	–(–)	–/–	–(–)/–(–)	–/**92.2**[**145**]	–(–)/**1.000**(**3794**)
MNIST@5-2	**98.9/97.1**[225]	**0.262(5754)**	96.9/93.7[256]	1.082(11022)	–/92.5[**218**]	–(–)/0.246(11512)	95.6[196]/95.8[166]	0.490(8227)/0.570(5179)
MNIST@5-3	98.8/95.7[126]	**0.383**(4212)	94.8/90.4[156]	2.085(10291)	96.8[106]/95.1[231]	0.689(11061)/0.770(13373)	97.6[226]/92.3[106]	0.680(10585)/1.000(**3617**)
MNIST@5-4	–	–(–)	92.9/91.7[166]	2.084(7087)	**97.1**[**121**]/96.1[61]	0.691(12028)/**0.177**(**4569**)	–/96.1[185]	–(–)/1.660(6057)
MNIST@5-5	**98.6/94.2**[255]	**0.208**(8649)	94.0/91.4[191]	0.774(16215)	95.2[241]/94.4[161]	0.475(25096)/0.163(12725)	95.7[**151**]/91.7[187]	0.340(7629)/0.500(**6580**)
MNIST@10-1	97.2/97.6[**161**]	**0.336(7196)**	92.3/91.8[196]	1.987(9277)	96.2[221]/96.8[201]	1.055(9651)/0.492(8796)	**98.0**[166]/95.3[187]	0.910(13762)/1.410(10479)
MNIST@10-2	**97.1/97.4**[**164**]	**0.523**(9277)	91.6/88.2[251]	4.354(11950)	97.0[201]/96.9[186]	1.587(9942)/0.753(**9212**)	95.6[181]/–	1.640(16238)/–(–)
MNIST@10-3	95.8/93.6[185]	**0.167**(10557)	95.1/90.6[221]	0.966(10820)	95.3[181]/95.3[181]	0.372(**9081**)/0.191(9107)	**98.1**[211]/94.03[201]	0.490(15360)/0.650(10167)
MNIST@10-4	**97.9/98.2**[**185**]	**0.167**(10292)	94.3/93.7[231]	1.001(13368)	97.3[161]/97.2[194]	0.331(**6472**)/0.202(7838)	–/–	–(–)/–(–)
MNIST@10-5	**98.4/97.4**[**115**]	**0.172(5786)**	96.1/95.7[167]	1.094(10012)	96.8[221]/96.3[141]	0.706(10432)/0.23(6734)	96.6[165]/97.04[165]	0.600(11001)/0.800(7670)
CIFAR-10@5-1	74.5/68.0[140]	**0.157(5649)**	–	–(–)	69.1[176]/66.4[146]	0.184(7269)/0.169(11628)	68.5[91]/64.9[106]	0.251(5825)/0.3(4868)
CIFAR-10@5-2	**79.1/83.5**[**150**]	**0.19**(7592)	74.3/68.4[241]	1.252(19315)	78.3[203]/–	0.244(10344)/–(–)	–/71.8[96]	–(–)/0.410(**4984**)
CIFAR-10@5-3	**72.1/73.3**[**96**]	**0.171(5031)**	68.5/67.4[176]	1.351(10762)	67.2[251]/64.1[171]	0.422(12929)/0.318(5919)	72.1[144]/72[108]	0.652(10329)/0.634(5979)
CIFAR-10@5-4	**76.8/73.1**[176]	**0.231**(8000)	68.9/72.3[216]	1.267(12585)	75.8[209]/67.8[200]	0.25(10972)/0.27(6741)	69.7[**76**]/65.7[96]	0.243(5754)/0.413(**5662**)
CIFAR-10@5-5	69.8/70.5[**70**]	**0.178(4492)**	–	–(–)	–/59.5[184]	–(–)/0.479(6734)	**75.1**[121]/69.8[121]	0.759(9594)/1(7247)
CIFAR-10@10-1	69.1/74.9[**151**]	**0.161(4354)**	61.2/60.3[161]	0.754(11273)	73.6[206]/–	0.205(15233)/–(–)	65.5[200]/66.1[**151**]	0.527(10233)/0.519(5990)
CIFAR-10@10-2	76.2/72.8[126]	**0.151(4356)**	67.9/68.4[221]	1.186(12781)	–/67.6[144]	–(–)/0.173(8704)	**77.5**[**82**]/70.9[202]	0.247(5254)/0.784(5772)
CIFAR-10@10-3	75.6/72.2[176]	**0.16(5684)**	69.1/64.7[**166**]	0.662(9983)	74.9[217]/66.3[211]	0.198(18887)/0.183(12733)	65.7[211]/55.9[173]	0.469(10138)/0.517(7748)
CIFAR-10@10-4	**71.1/69.3**[**114**]	**0.232(3990)**	64.6/67.8[246]	2.245(11128)	–/65.8[231]	–(–)/0.497(10239)	–/62.7[160]	–(–)/1.102(4165)
CIFAR-10@10-5	75.5/76.9[96]	0.177(**3313**)	70.53/68.8[201]	1.598(11524)	**81.9**[104]/75.1[131]	**0.176**(9513)/0.252(7045)	76.2[**92**]/74.5[158]	0.423(7808)/0.900(6284)
CIFAR-100@5-1	**53.4/49.3**[**246**]	**0.212(5125)**	44.7/47.9[**246**]	0.84(7727)	46.7[261]/47.6[266]	0.224(6080)/0.250(5215)	48.4[251]/48.5[276]	0.517(6207)/0.732(5389)
CIFAR-100@5-2	46.3/44.0[256]	**0.244**(5949)	42.1/44.9[**171**]	0.645(3261)	45.7[266]/42.3[251]	0.253(6522)/0.26(4575)	45.7[226]/38.8[226]	0.509(7598)/0.658(**4178**)
CIFAR-100@5-3	47.4/57.7[**181**]	**0.187(4499)**	45.6/51.8[191]	0.682(4234)	49.2[231]/46.9[271]	0.247(5518)/0.281(5403)	48.9[251]/46.3[241]	0.572(8801)/0.701(5263)
CIFAR-100@10-1	39.9/45.6[241]	**0.225**(9400)	39.1/42.8[276]	1.02(10524)	43.5[256]/40.5[256]	0.238(10666)/0.26(8023)	44.4[**216**]/43.7[271]	0.467(**8002**)/0.772(8447)
CIFAR-100@10-2	43.5/37.7[246]	**0.207**(8285)	41.4/34.9[281]	1.004(11676)	39.0[266]/37.4[**236**]	0.233(8951)/0.225(**8075**)	38.0[261]/33.3[276]	0.546(11404)/0.747(9901)
CIFAR-100@10-3	51.8/43.1[231]	**0.200**(9390)	48.7/41.1[246]	0.787(12057)	46.0[221]/42.5[241]	0.207(10938)/0.217(**8107**)	44.4[266]/42.6[**226**]	0.527(10635)/0.573(8691)
CIFAR-100@20-1	28.6/32.3[256]	0.239(40916)	26.3/30.9[266]	0.888(44196)	29.3[251]/31.1[176]	0.219(33272)/**0.168**(**22880**)	29.7[256]/31.0[276]	0.531(41338)/0.747(46047)
CIFAR-100@20-2	27.1/29.7[**221**]	**0.199**(40345)	25.5/24.3[274]	0.91(42624)	27.4[236]/29.7[241]	0.202(**30081**)/0.226(37947)	26.3[271]/27.3[261]	0.552(43016)/0.693(37622)

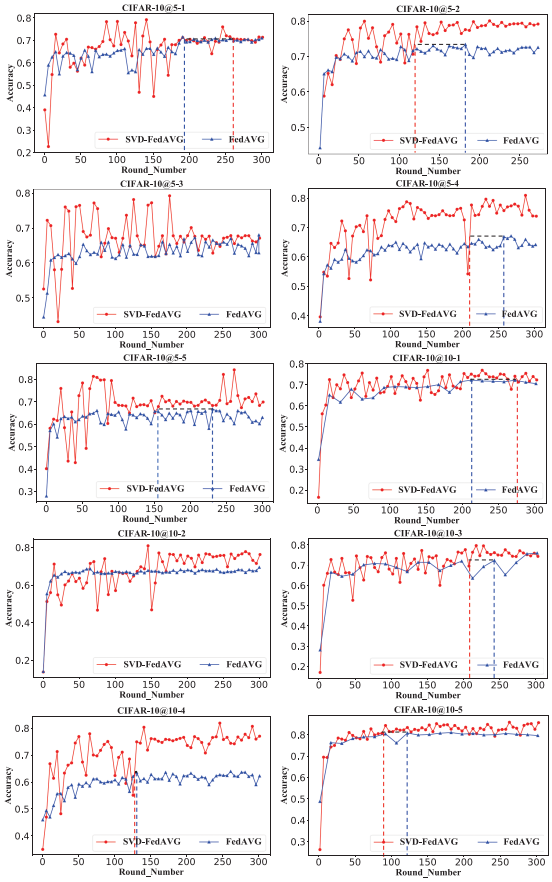


Fig. 5. Comparative studies of convergence speed on SVD-FedAVG and FedAVG on CIFAR-10 dataset.

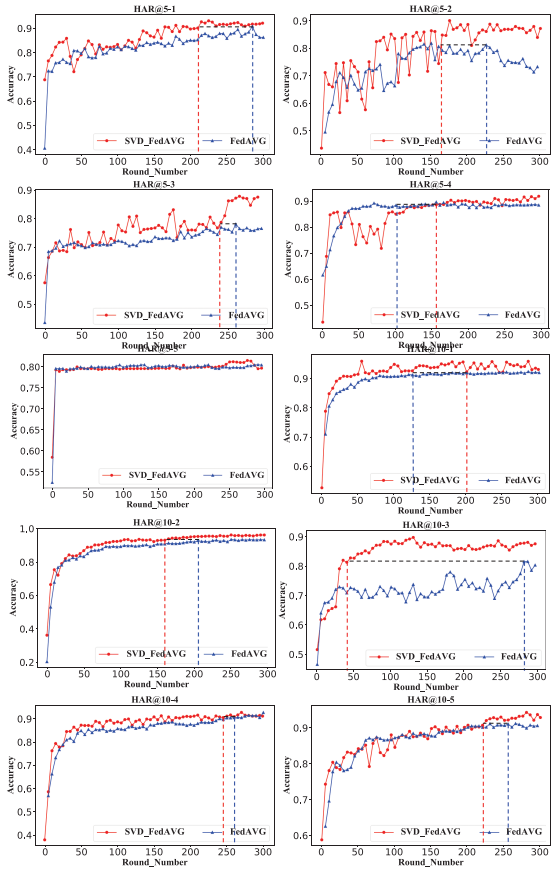


Fig. 6. Comparative studies of convergence speed on SVD-FedAVG and FedAVG on HAR dataset.

lowest due to the smallest scale of uploading data in each single round. However, the costs of the SVD-HCMFL framework is lower than SVD-FedAVG in almost all cases. (3) If we refer back to the results of Table V, we can find the performance of SVD-HCMFL is better than HCMFL and SVD-FedAVG, demonstrating the contributions of HCMFL and SVD-FedAVG can mutually enhance each other. Additionally, we also observe that FedLR-HCMFL outperforms FedLR in most cases, which

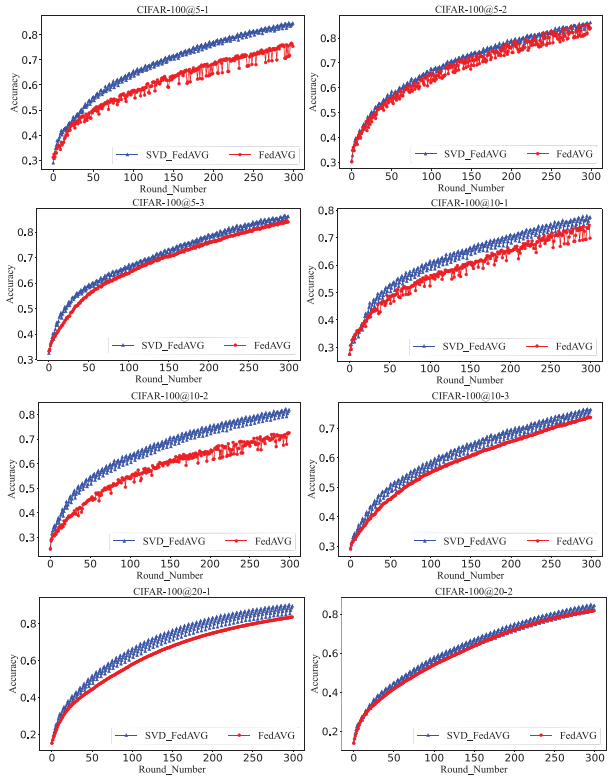


Fig. 7. Visualization of training accuracy w.r.t the communication rounds on CIFAR-100 dataset.

TABLE VII
PARAMETER SETTINGS OF SUBORDINATED CNNS

Type	Layer (Activation Function)	Parameter Size
S_2	conv2d_1(relu)	[3×3×3+1]×32
	conv2d_2(relu)	[3×3×32+1]×32
	dense_1(relu)	[8192+1]×128
	dense_2(softmax)	[128+1]×100
D_2	conv2d_1(relu)	[3×3×3+1]×32
	conv2d_2(relu)	[3×3×32+1]×32
	conv2d_3(relu)	[3×3×32+1]×32
	conv2d_4(relu)	[3×3×32+1]×64
	conv2d_5(relu)	[3×3×64+1]×64
	conv2d_6(relu)	[3×3×64+1]×64
	dense_1(relu)	[1600+1]×512
	dense_2(softmax)	[512+1]×10

TABLE VIII
CONVERGENCE ACCURACY OF SVD-HCMFL WITH FOUR HETEROGENEOUS STRUCTURED GLOBAL MODELS

Methods	Models	CIFAR-100@20-1	CIFAR-100@20-2
SVD-HCMFL	*DCNNs[D1/D2]*	30.17/29.27	29.41/27.93
	SCNNs[S1/S2]	32.27/30.79	29.49/27.69
ASTW_FedAVG	*DCNNs[D1/D2]*	29.7/29.57	28.3/26.79
	SCNNs[S1/S2]	31.0/30.66	27.3/27.43
FedAVG	*DCNNs[D1/D2]*	28.9/29.90	25.4/26.17
	SCNNs[S1/S2]	32.2/30.14	27.8/25.22

illustrates the superiority of FL with heterogeneous structured models. Summarily, these results thoroughly illustrate the effectiveness of combining the heterogeneous FL with the efficient communication strategy.

Our study aims to obtain heterogeneous structured global models by aggregating local model parameters from different clients. In order to illustrate the effectiveness of FL, we compare the performance of the heterogeneous structured global models with the results achieved by heterogeneous structured models that do not connect to the server. The results of models without connecting to the server are presented in the column of "Heterogeneous Mode" in Table III. We can observe 0.69% and 2.42% along with 2.73% and 1.34% of accuracy promotions for DCNN and SCNN in the MNIST and CIFAR-10 datasets, respectively. It is worth noting the results of models in heterogeneous mode can not represent the generalization ability of local models. In other words, some clients with scarce training data may still struggle to get a powerful learning model under such cases. In contrast, the global model can help all clients to improve learning performance by exploiting the knowledge from each other.

The SVD operation in SVD-HCMFL reduces the communication costs but inevitably increases the extra overhead of each client. As the communication costs and computation complexity are measured in different criteria, we take the time costs for global model convergence as measurements to compare the resource consumption. All experimental settings, including the bandwidth and computing resources, are kept identical for fair comparisons. According to the results, the time consumption of SVD-HCMFL is generally lower than those of comparing methods. On three datasets, SVD-HCMFL saves 677 s, 1544 s, 4237 s, and 1605 s compared with FedAVG, HCMFL, FedLR-HCMFL, and ASTW_FedAVG algorithms, respectively. In our experiments, the environment of FL is simulated within local area network, the communication distance in our simulation can be neglected. That is to say, the reduction on time consumption can be more obvious when it comes to the communication between some distant institutions.

The above experiments are based on the threshold achieved in Section IV-B1. To mitigate the effects of specific threshold values, we design two subordinated models inside the DCNNs and SCNNs, respectively. That is to say, there are four global models with heterogeneous structures on the server. The DCNN and SCNN introduced in Section IV-A are indexed as S_1 and D_1, respectively. Specific model settings of another two models are shown in Table VII. These subordinated models are randomly assigned to different clients according to the previously threshold. In specific, the clients with data volume above the threshold randomly select the subordinated model inside DCNNs. Specific results are presented in Table VIII. We can observe that the heterogeneous structured models generally outperform the ASTW_FedAVG and FedAVG, which utilize homogeneous models. The promotions demonstrate the superiority of adopting heterogeneous structured models over the using of global models with identical structures, regardless of the specific threshold values. The results also demonstrate the feasibility of our framework with more than two heterogeneous structured global models.

TABLE IX
THE AMOUNT OF DATA STORED IN DIFFERENT CLIENTS

Sensor#	Size of sampled dataset
Sensor_1	14723
Sensor_2	21709
Sensor_3	19064
Sensor_4	9957
Sensor_5	14565
Sensor_6	17170

C. Application of SVD-HCMFL in the Trend Following of Electromagnetic Radiation Intensity From Coal Mine

The safety forecast of coal and gas outburst dynamic disasters is generally based on the variation of electromagnetic radiation intensity time series data (ERI-TSD) collected from the underground mine [40]. Precise trend following of ERI-TSD is thus essential for the safety production. In previous research, we had already given the corresponding trend representations and an improved LSTM based single-sensor-oriented trend following mechanism in [41], and further studied the FL for multi-sensor data fusion in [42]. Results have shown that the trend following accuracy is greatly promoted with FL, but the communication cost is quite high. Here, we apply the proposed SVD-HCMFL framework to this problem and expect to further improve the performance of trend following of ERI-TSD with lower communication cost.

The average ($A_{r,l}$) and the standard deviation ($\sigma_{r,l}$) representations defined in [41] are used as the trend expression. Time-series data from 6 different sensors are collected for study. The amount of each sensor is quite different and the details are given in Table IX. Sensors are considered as different clients in the FL framework. The LSTM and GRU (Gated Recurrent Unit) networks [43] are set on the central server for different local clients. Compared with the LSTM network, the GRU network with fewer feature extraction layer parameters usually requires smaller training set, so the two can be taken as a deep network and a shallow one. Sensor_1, Sensor_4 and Sensor_5 take the GRU models and the rest selects the LSTM. As for the structure of LSTM, there are one lstm_layer with $cell\_size$=5, $time\_step$=5 and two dense layers with 64 units and 1 output. The structure of GRU is the same with the structure of LSTM except for the gru_layer. Sliding window method is applied to the prediction, and the step is set to 5.

We combine the SVD-HCMFL framework with the method proposed in [42]. Each client uses the downloaded LSTM/GRU network to extract the features of the local dataset. Whereafter, the extracted features and the real values of local data are fed into Echo State Networks (ESNs) [43]. Finally, these ESNs are used to calculate the corresponding trend volatility for trend following. The number of communication rounds and the agg_round are respectively set to 10 and 5 according to [42].

The accuracies of the SVD-HCMFL based method (abbreviated to SVD-HCM), the HCMFL based method, and aggregated features based method (Fea_based for short) [42] are compared. Four indicators, i.e., root relative squared error (RRSE), root mean square error (RMSE), mean absolute percent error (MAPE), and maximum error (Max-error) are used to measure the accuracy. The communication cost (C.cost) is adopted to evaluate the communication efficiency. The trend following results of $A_{r,l}$ and $\sigma_{r,l}$ with different methods are presented in Figs. 8 and 9, while the specific indicators are shown in Tables X and XI, respectively. Accordingly, we observe that: (1) For local clients with different scale data sizes, the accuracy of trend following based on heterogeneous structured FL is higher than the traditional algorithms using a single central model-based method. (2) The method proposed in [42] requires the highest communication costs but obtains lowest accuracy. These experimental results further illustrate the effectiveness of our proposed framework in solving practical regression problems.

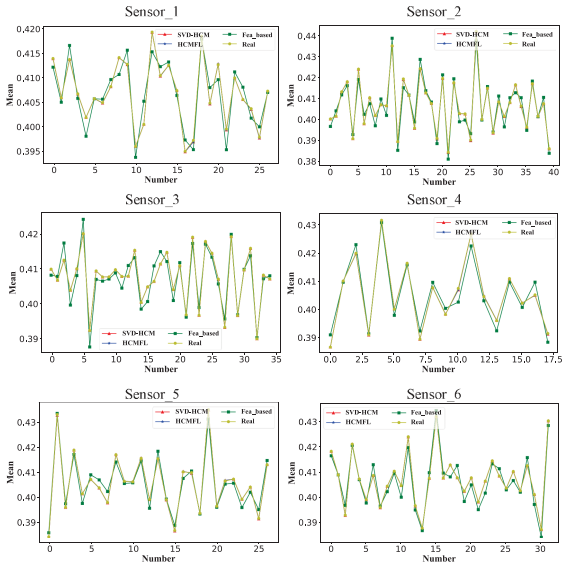


Fig. 8. Trend following results of $A_{r,l}$ on each individual Client.

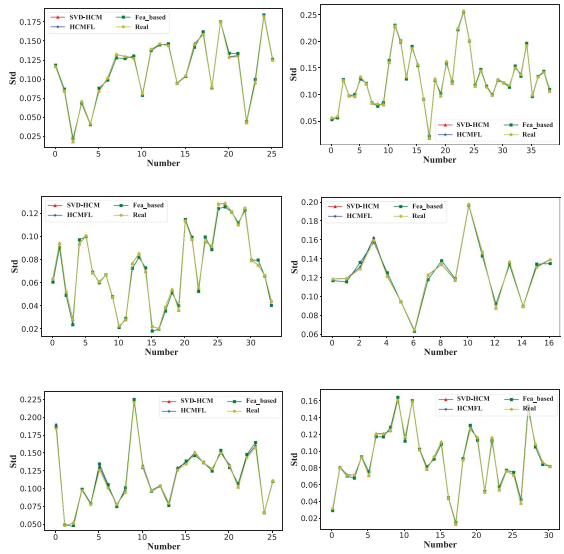


Fig. 9. Trend following results of $\sigma_{r,l}$ on each individual Client.

TABLE X
SPECIFIC INDICATORS OF TREND FOLLOWING RESULTS OF $A_{r,l}$ ON EACH INDIVIDUAL CLIENT

Sensor#	Methods	RRSE	RMSE	MAPE	Max-error	C.cost
Sensor_1	SVD-HCM	**0.0152**	**0.00009**	**0.0002**	**0.0003**	**0.431**
	HCMFL	0.017	0.0001	**0.0002**	0.0004	1
	Fea_based	0.3764	0.0024	0.0052	0.0047	29.2
Sensor_2	SVD-HCM	0.008	0.0001	0.0002	**0.0003**	**0.465**
	HCMFL	**0.0065**	**0.00008**	**0.0001**	**0.0003**	1
	Fea_based	0.2445	0.00306	0.0066	0.0051	29.2
Sensor_3	SVD-HCM	**0.003**	**0.00003**	**0.00004**	**0.00005**	**0.465**
	HCMFL	0.004	**0.00003**	0.00005	0.00007	1
	Fea_based	0.3296	0.00249	0.0051	0.0049	29.2
Sensor_4	SVD-HCM	0.0129	0.00016	0.00028	0.0005	**0.431**
	HCMFL	**0.0117**	**0.00014**	**0.00025**	**0.0003**	1
	Fea_based	0.2284	0.0028	0.0059	0.0048	29.2
Sensor_5	SVD-HCM	**0.0103**	**0.0001**	**0.0002**	**0.0003**	**0.431**
	HCMFed	0.0412	0.0005	0.00063	0.0021	1
	Fea_based	0.1979	0.0023	0.0048	0.0045	29.2
Sensor_6	SVD-HCM	**0.0046**	**0.00005**	**0.00008**	**0.0001**	**0.465**
	HCMFL	0.0086	0.0003	0.0025	0.0007	1
	Fea_based	0.2827	0.0029	0.0061	0.0049	29.2

TABLE XI
SPECIFIC INDICATORS OF TREND FOLLOWING RESULTS OF $\sigma_{r,l}$ ON EACH INDIVIDUAL CLIENT

Sensor#	Methods	RRSE	RMSE	MAPE	Max_error	C.cost
Sensor_1	SVD-HCM	**0.0127**	**0.0005**	0.0036	0.0013	**0.431**
	HCMFL	0.0157	0.0006	**0.0002**	**0.0004**	1
	Fea_based	0.0728	0.0029	0.0304	0.0056	29.2
Sensor_2	SVD-HCM	0.0139	**0.0007**	0.0037	**0.0016**	**0.465**
	HCMFL	**0.0134**	**0.0007**	**0.0035**	0.0017	1
	Fea_based	0.0569	0.0028	0.0256	0.0055	29.2
Sensor_3	SVD-HCM	**0.0037**	**0.0001**	**0.0011**	0.0004	**0.465**
	HCM	0.0047	0.0002	0.0013	**0.0003**	1
	Fea_based	0.0888	0.0028	0.0433	0.0047	29.2
Sensor_4	SVD-HCM	**0.0326**	**0.001**	**0.0041**	0.0029	**0.431**
	HCMFL	**0.0326**	**0.001**	0.0055	**0.0025**	1
	Fea_based	0.1158	0.0034	0.0233	0.007	29.2
Sensor_5	SVD-HCM	**0.0543**	**0.0021**	**0.0118**	**0.006**	**0.431**
	HCM	0.0765	0.003	0.0177	0.0075	1
	Fea_based	0.0979	0.0038	0.0274	0.0089	29.2
Sensor_6	SVD-HCM	0.0096	**0.0003**	0.0028	0.0008	**0.465**
	HCMFL	**0.0086**	**0.0003**	**0.0025**	**0.0007**	1
	Fea_based	0.0862	0.0031	0.0353	0.0051	29.2

V. CONCLUSION

Since canonical FL does not consider differences among various local datasets, this paper has investigated the superiority of heterogeneous structured models over the homogeneous model in FL. An SVD-based communication efficient mechanism has been presented to further reduce the communication costs. The proposed framework has been applied to four different image classification tasks, and the multi-sensor collected ERI-TSD regression task. The results demonstrate the proposed methods can effectively improve the convergence accuracy and stability, meanwhile reducing the communication costs. The current study only adopts different predefined heterogeneous models to validate the effectiveness of our framework. In future work, we expect to further investigate adaptive model structure designs according to the characteristics of local samples, e.g., via neural architecture search. In addition, our future work also includes the study of FL for client-side heterogeneous data such as text, images, and videos.

REFERENCES

[1] Y.-S. Ong and A. Gupta, "AIR 5: Five pillars of artificial intelligence research," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 3, no. 5, pp. 411–415, Oct. 2019.

[2] Y. Shi, C.-T. Lin, Y.-C. Chang, W. Ding, Y. Shi, and X. Yao, "Consensus learning for distributed fuzzy neural network in Big Data environment," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 5, no. 1, pp. 29–41, Feb. 2021.

[3] M. Asim, Y. Wang, K. Wang, and P.-Q. Huang, "A review on computational intelligence techniques in cloud and edge computing," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 4, no. 6, pp. 742–763, Dec. 2020.

[4] H. 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.

[5] J. Konečnỳ, B. McMahan, and D. Ramage, "Federated optimization: Distributed optimization beyond the datacenter," in *Proc. NIPS Workshop Optim. Mach. Learn.*, 2016. [Online]. Available: https://arxiv.org/pdf/1511.03575

[6] 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.

[7] K. Bonawitz et al., "Practical secure aggregation for privacy-preserving machine learning," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2017, pp. 1175–1191.

[8] Y. Chen, X. Sun, and Y. Hu, "Federated learning assisted interactive EDA with dual probabilistic models for personalized search," in *Proc. Int. Conf. Swarm Intell.*, 2019, pp. 374–383.

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

[10] Y. Deng, "Deep learning on mobile devices: A review," in *Proc. Mobile Multimedia/Image Process., Secur., Appl.*, Bellingham, WA, USA: SPIE, vol. 10993, 2019, pp. 52–66.

[11] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.

[12] A. Kolesnikov et al., "Big transfer (bit): General visual representation learning," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 491–507.

[13] Z. Cao, L. Ma, M. Long, and J. Wang, "Partial adversarial domain adaptation," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 135–150.

[14] D. L. Pham, C. Xu, and J. L. Prince, "A survey of current methods in medical image segmentation," *Annu. Rev. Biomed. Eng.*, vol. 2, no. 3, pp. 315–337, 2000.

[15] M. Aladem and S. A. Rawashdeh, "A single-stream segmentation and depth prediction CNN for autonomous driving," *IEEE Intell. Syst.*, vol. 36, no. 4, pp. 79–85, Jul./Aug. 2021.

[16] H. Zhu and Y. Jin, "Multi-objective evolutionary federated learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 4, pp. 1310–1322, Apr. 2020.

[17] H. Zhu and Y. Jin, "Real-time federated evolutionary neural architecture search," *IEEE Trans. Evol. Comput.*, vol. 26, no. 2, pp. 364–378, Apr. 2022.

[18] J. Konecný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," in *Proc. NIPS Workshop Private Multi-Party Mach. Learn.*, 2016. [Online]. Available: https://arxiv.org/abs/1610.054

[19] Y. Mao et al., "Communication efficient federated learning with adaptive quantization," *ACM Trans. Intell. Syst. Technol.*, vol. 13, pp. 1–26, 2022.

[20] G. Yan, S.-L. Huang, T. Lan, and L. Song, "DQSGD: Dynamic quantized stochastic gradient descent for communication-efficient distributed learning," in *Proc. IEEE 18th Int. Conf. Mobile Ad Hoc Smart Syst.*, 2021, pp. 136–144.

[21] M. Yurochkin, M. Agarwal, S. Ghosh, K. Greenewald, T. N. Hoang, and Y. Khazaeni, "Bayesian nonparametric federated learning of neural networks," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 7252–7261.

[22] M. Abadi et al., "Deep learning with differential privacy," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2016, pp. 308–318.

[23] R. C. Geyer, T. Klein, and M. Nabi, "Differentially private federated learning: A client level perspective," in *Proc. NIPS Workshop Mach. Learn. Phone other Consum. Devices*, 2017. [Online]. Available: https://arxiv.org/pdf/1712.07557v1

[24] Y. Jiang, J. Konečnỳ, K. Rush, and S. Kannan, "Improving federated learning personalization via model agnostic meta learning," in *Proc. Int. Workshop Federated Learn. User Privacy Data Confidentiality Conjunction NeurIPS*, 2019. [Online]. Available: https://arxiv.org/pdf/1909.12488

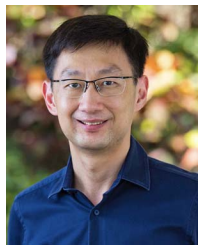
[25] X. Peng, Z. Huang, Y. Zhu, and K. Saenko, "Federated adversarial domain adaptation," in *Proc. Int. Conf. Learn. Representations*, 2020, pp. 1–19.
[26] J. R. Bunch and J. E. Hopcroft, "Triangular factorization and inversion by fast matrix multiplication," *Math. Comput.*, vol. 28, no. 125, pp. 231–236, 1974.
[27] E. Anderson, Z. Bai, and J. Dongarra, "Generalized QR factorization and its applications," *Linear Algebra Appl.*, vol. 162, pp. 243–271, 1992.
[28] O. Schenk and K. Gärtner, "Solving unsymmetric sparse systems of linear equations with pardiso," *Future Gener. Comput. Syst.*, vol. 20, no. 3, pp. 475–487, 2004.
[29] J. Ye and Q. Li, "A two-stage linear discriminant analysis via QR-decomposition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 6, pp. 929–941, Jun. 2005.
[30] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemometrics Intell. Lab. Syst.*, vol. 2, no. 1/3, pp. 37–52, 1987.
[31] N. K. Kumar and J. Schneider, "Literature survey on low rank approximation of matrices," *Linear Multilinear Algebra*, vol. 65, no. 11, pp. 2212–2244, 2017.
[32] H. Kim, M. U. K. Khan, and C.-M. Kyung, "Efficient neural network compression," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 12569–12577.
[33] Y. Idelbayev and M. A. Carreira-Perpinán, "Low-rank compression of neural nets: Learning the rank of each layer," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 8049–8059.
[34] K. Goetschalckx, B. Moons, P. Wambacq, and M. Verhelst, "Efficiently combining SVD, pruning, clustering and retraining for enhanced neural network compression," in *Proc. 2nd Int. Workshop Embedded Mobile Deep Learn.*, 2018, pp. 1–6.
[35] H. Zhou, J. Cheng, X. Wang, and B. Jin, "Low rank communication for federated learning," in *Proc. Int. Conf. Database Syst. Adv. Appl.*, 2020, pp. 1–16.
[36] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "LSTM: A search space odyssey," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 10, pp. 2222–2232, Oct. 2017.
[37] F. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with LSTM," in *Proc. 9th Int. Conf. Artif. Neural Netw.*, 1999, pp. 850–855.
[38] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of fedavg on non-IID data," in *Proc. Int. Conf. Learn. Representations*, 2019, pp. 1–26.
[39] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz, "A public domain dataset for human activity recognition using smartphones," in *Proc. Eur. Symp. Artif. Neural Netw.*, 2013, pp. 437–442.
[40] E. Wang, Z. Li, X. He, and C. Liang, "Application and pre-warning technology of coal and gas outburst by electromagnetic radiation," *Coal Sci. Technol.*, vol. 42, no. 6, pp. 53–57, 2014.
[41] Y. Hu, X. Sun, X. Nie, Y. Li, and L. Liu, "An enhanced LSTM for trend following of time series," *IEEE Access*, vol. 7, pp. 34020–34030, 2019.
[42] Y. Hu, X. Sun, Y. Chen, and Z. Lu, "Model and feature aggregation based federated learning for multi-sensor time series trend following," in *Proc. Int. Work- Conf. Artif. Neural Netw.*, 2019, pp. 233–246.
[43] H. Jaeger and H. Haas, "Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication," *Science*, vol. 304, no. 5667, pp. 78–80, 2004.

Yao Hu received the B.S. degree in mining engineering and the M.Sc. degree in control science and engineering from the China University of Mining and Technology, Xuzhou, China, in 2017 and 2020, respectively. He is currently working toward the Ph.D. degree with the Department of Computer Science, City University of Hong Kong, Hong Kong. His research interests include machine learning, transfer learning, and federated learning.

Xiaoyan Sun (Member, IEEE) received the Ph.D. degree in control theory and control engineering from the China University of Mining and Technology, Xuzhou, China, in 2009. She is currently a Professor with the School of Information and Control Engineering, China University of Mining and Technology. Her research interests include interactive evolutionary computation, Big Data, and intelligence optimization.

Ye Tian (Member, IEEE) received the B.Sc., M.Sc., and Ph.D. degrees from Anhui University, Hefei, China, in 2012, 2015, and 2018, respectively. He is currently an Associate Professor with the Institutes of Physical Science and Information Technology, Anhui University, Hefei, China. His research interests include evolutionary computation and its applications. He was the recipient of the 2018 and 2021 IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION Outstanding Paper Award, 2020 IEEE Computational Intelligence Magazine Outstanding Paper Award, and 2022 IEEE Computational Intelligence Society Outstanding Ph.D. Dissertation Award.

Linqi Song (Senior Member, IEEE) received the B.S. and M.S. degrees from Tsinghua University, Beijing, China, and the Ph.D. degree from the University of California at Los Angeles, Los Angeles, CA, USA. He is currently an Assistant Professor with the Department of Computer Science, City University of Hong Kong, Hong Kong, and the Research Scientist with the City University of Hong Kong, Shenzhen Research Institute, Shenzhen, China. His research interests include information theory, machine learning, and Big Data. He was recipient of the Hong Kong RGC Early Career Scheme in 2019 and the Best Paper Award from IEEE MIPR 2020.



Kay Chen Tan (Fellow, IEEE) received the B.Eng. degree (with First Class Hons.) and the Ph.D. degree from the University of Glasgow, Glasgow, U.K., in 1994 and 1997, respectively. He is currently a Chair Professor with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong. He was the Editor-in-Chief of IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, and is currently the Editorial Board Member of more than ten journals. He is also the Vice-President (Publications) of IEEE Computational Intelligence Society, IEEE Distinguished Lecturer Program (DLP) speaker, Honorary Professor with the University of Nottingham, Nottingham, U.K., and the Chief Co-Editor of Springer Book Series on *Machine Learning: Foundations, Methodologies, and Applications*.