












# FedHM: Efficient federated learning for heterogeneous models via low-rank factorization

Dezhong Yao <sup>a, \*</sup>, Wanning Pan <sup>a, </sup>, Yuexin Shi <sup>a, </sup>, Michael J. O'Neill <sup>b, </sup>,  
Yutong Dai <sup>c, </sup>, Yao Wan <sup>a, </sup>, Peilin Zhao <sup>d, </sup>, Hai Jin <sup>a, </sup>, Lichao Sun <sup>c, </sup>

<sup>a</sup> National Engineering Research Center for Big Data Technology and System, Services Computing Technology and System Lab, Cluster and Grid Computing Lab, School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, China

<sup>b</sup> University of North Carolina, Chapel Hill, USA

<sup>c</sup> Lehigh University, USA

<sup>d</sup> Tencent AI Lab, China

## ARTICLE INFO

### Keywords:

Federated learning  
Low-rank factorization  
Model heterogeneity

## ABSTRACT

One underlying assumption of recent *Federated Learning* (FL) paradigms is that all local models share an identical network architecture. However, this assumption is inefficient for heterogeneous systems where devices possess varying computation and communication capabilities. The presence of such heterogeneity among devices negatively impacts the scalability of FL and slows down the training process due to the existence of stragglers. To this end, this paper proposes a novel *federated compression framework for heterogeneous models*, named FedHM, distributing the heterogeneous low-rank models to clients and then aggregating them into a full-rank global model. Furthermore, FedHM significantly reduces communication costs by utilizing low-rank models. Compared with state-of-the-art heterogeneous FL methods under various FL settings, FedHM is superior in the performance and robustness of models with different sizes. Additionally, the convergence guarantee of FL for heterogeneous devices is first theoretically analyzed.

## 1. Introduction

*Federated Learning* (FL) [1] is a collaborative framework that enables multiple clients (devices) to train a global model while preserving data privacy on each device. FL is widely adopted for distributed training on IoT devices due to its privacy preservation and communication efficiency compared to transmitting local data to the cloud server [2–4]. Despite these advantages, several practical challenges still need to be addressed [5–7], which requires a rethink of existing frameworks. Among these challenges, one critical issue is system heterogeneity, which arises from the presence of devices with slow processing capabilities, commonly referred to as *straggling devices*. These straggling devices significantly slow down model convergence [5,8].

This paper focuses on system heterogeneity in FL, where devices may possess varying computing capabilities and communication capabilities due to hardware disparities [8–11]. When faced with system heterogeneity, it is impractical to expect all clients to train models with the same architecture [12,13]. As illustrated in Fig. 1, consider a scenario in which a smartphone equipped with a CPU

\* Corresponding author.

E-mail address: [dyao@hust.edu.cn](mailto:dyao@hust.edu.cn) (D. Yao).

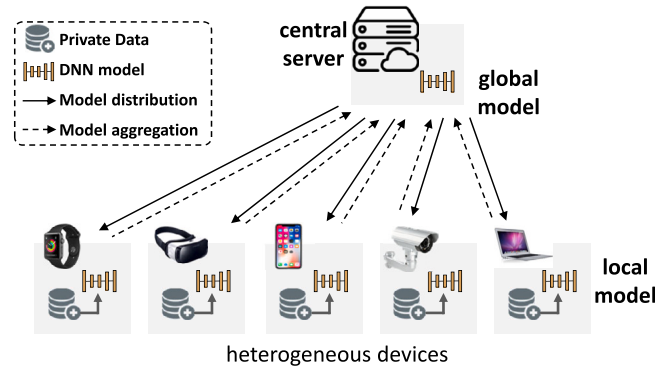


Fig. 1. The FL framework for heterogeneous computation scenario. The locally trained models on each device share the same model architecture, which is inefficient for resource-constrained devices.

and a laptop equipped with a GPU participate in FL. Deploying identical models will slow down the training process due to variations in computing capacity. On the one hand, using a small model will prevent the laptop from fully utilizing its computational resources. On the other hand, deploying a large model may render the smartphone incapable of participating in FL due to limited resources. Furthermore, clients with weak communication capabilities or those with weak network signals need to transfer fewer parameters than other clients. Due to the above reasons, addressing system heterogeneity requires training heterogeneous models that align with the hardware resources of each client, which introduces the challenge *model heterogeneity*.

Currently, several approaches are being employed to address the model heterogeneity problem in FL. FedMD [14], FedDF [15], DS-FL [16] and Fed-ET [17] attempt to aggregate heterogeneous local models based on knowledge distillation (KD) [18]. However, the performance of these methods depends on the size of the proxy data and the domain similarity between the proxy data and the local data [19]. HeteroFL [20] adopts uniform channel pruning (width slimming) for each local model and aggregates them by channels. However, this static extraction strategy prevents certain parts of the global model from being trained on client devices with limited resources. Consequently, different parts of the global model are trained on data with varying distributions, resulting in performance degradation of the global model. Split-Mix [21] and FedRox [22] are proposed to solve this problem. Split-Mix extracts a sub-model pool from the global model and randomly selects one or several sub-models to form a local model. FedRox utilizes a rolling sub-model extraction scheme that ensures different parts of the global model are evenly trained. However, these *partial training* (PT)-based strategies are unable to train all parameters of the global model fairly during each communication round. In other words, different parts of the global model cannot be trained on data with different distributions during each communication round, leading to a performance bottleneck. Besides, none of these methods provides a convergence guarantee.

Given the limitations of existing approaches, we aim to overcome the constraints imposed by KD and PT. Compared to KD, without relying on proxy data, we propose a novel Federated model compression strategy for Heterogeneous Models, called FedHM, to further improve the performance of model-heterogeneous FL. FedHM enables multiple devices with varying computation and communication capabilities to train heterogeneous local models. Moreover, it ensures that all global model parameters can be updated while training the various local models, thus completely resolving the drawbacks caused by PT. Split-Mix and FedRox can only update some of the parameters in one round of local training on constrained clients, while FedHM can update all parameters, even on the weakest clients. Our key idea is to compress a large model into small models using low-rank factorization and distribute the compressed models to resource-constrained devices for local training. Specifically, FedHM applies DNN factorization [23] to create a hybrid network architecture that maintains the capability of the model, which means some layers are low-rank factorized and other layers are kept full rank. The server then collects the trained heterogeneous local models from the selected clients and transforms each local model back into its original full-rank model shape. Subsequently, the global model is easily aggregated from these transformed local models. In particular, the factorization of the model is computed on the server, thus not imposing additional computation overhead on the clients. Additionally, motivated by the success of parameter-efficient adaptation [24–26], we propose a Federated learning Low-Rank Adaptation method for tuning pre-trained models, named FedLoRA. FedLoRA trains the low-rank parameters, which not only reduces computational and communication costs, but also allows for more flexible task switching by the transformers. Overall, the main contributions of this paper are as follows:

- We propose FedHM, which is the first work that utilizes the low-rank factorization mechanisms of DNNs to solve the model heterogeneity challenge in FL. Our proposed method outperforms existing state-of-the-art methods for heterogeneous FL.
- We prove a convergence guarantee for FedHM in the model heterogeneous FL setting.
- Our approach achieves high communication efficiency and can be flexibly adapted to various heterogeneous scenarios, including different models such as CNNs and transformers, as well as various deep learning scenarios such as training from scratch and tuning pre-trained models.

## 2. Related work

**Model Heterogeneity in FL.** FL is a promising alternative for distributed training [6], allowing participants to collaboratively build a unified model without sharing their private training data [7]. Model heterogeneity is a common challenge in FL, which can be addressed by KD-based methods and PT-based methods. In KD-based methods, FedMD [14], FedDF [15], DS-FL [16], and Fed-ET [17] use proxy data to break the knowledge barriers between heterogeneous client models. FedGH [27] trains a shared generalized global prediction header by uploading the locally averaged representation and label for each class. However, the availability of proxy data is not always guaranteed, and the final global model performance depends on the size of the proxy data and its domain similarity with the local data [19]. To solve this problem, DENSE [28] and DFRL [29] use a local model to train an additional generator on the server side to generate the data required for KD. In PT-based methods, HeteroFL [20] and FjORD [30] incorporate ideas from [31] and trains networks with different widths between different clients. However, such static assignments lead to certain parameters of the global model being inadequately trained on the dataset. In response, FedRolex [22] employs a rolling sub-model extraction scheme to extract dynamic sub-models. Similarly, Split-Mix [21] extracts a sub-model pool from the global model and randomly selects one or several sub-models to form a local model.

**Low-rank Factorization.** Low-rank factorization is a widely used compression method commonly deployed [32–34]. The core idea behind low-rank factorization is to approximate large-weight matrices in neural networks by-products of smaller matrices, thereby reducing the number of parameters and operations required during training and inference. [23] demonstrate that by applying singular value decomposition (SVD) to the convolutional layers of deep networks, significant reductions in both the number of parameters and computational load could be achieved without substantial loss in accuracy. [32] propose a method using CP-decomposition to factorize the convolutional kernels into products of lower-dimensional tensors, thus reducing the computational cost of convolutional operations. Moreover, [35] introduced a block-term decomposition method, which combines both CP and Tucker decompositions for a more flexible factorization.

**Low-rank Factorization in FL.** Applying low-rank factorization to FL presents several challenges. On the one hand, it is impractical to factorize the model after training in FL [36,37]. On the other hand, directly training a low-rank model from scratch in FL leads to a decrease in performance [38]. FedDLR [39] factorizes the server model and recovers the low-rank model on the clients. However, this approach increases the computation cost involved. FedPara [40] and Factorized-FL [41] factorize the model parameters to solve data heterogeneity in personalized FL [42–44], but do not consider model heterogeneity between clients. As for the Transformer-based language models, FedTune [45] and FedPETuning [46] conduct experimental exploration on the application of Adapter Tuning [47] and Prefix Tuning [48] in FL. In this paper, we utilize low-rank factorization to handle the model heterogeneity in FL, and simultaneously reduce computational and communication burden.

## 3. Methodology

In this section, we first introduce the low-rank decomposition of fully connected networks and convolutional neural networks. Then, we provide an overall introduction to the FedHM algorithm framework we propose and describe each step in the FedHM algorithm process in detail.

### 3.1. Local factorized training

**Fully-Connected (FC) Layer Factorization.** In the  $l$ -th layer of a fully connected network, the input vector  $x_{l-1}$  has a dimension of  $a$ , and the layer outputs  $f(x) = \sigma(W_l x_{l-1})$  after performing matrix multiplication has a dimension of  $b$ . Here,  $\sigma$  denotes the activation function, and  $W_l$  represents the weight matrix of the  $l$ -th layer. In FedHM, we propose replacing the weight matrix  $W_l \in \mathbb{R}^{a \times b}$  with a low-rank composition  $U_l V_l^T$ , where  $U_l \in \mathbb{R}^{b \times r}$ ,  $V_l \in \mathbb{R}^{a \times r}$ , and  $r \ll \min(a, b)$ . By training low-rank matrices  $U_l$  and  $V_l$  instead of  $W_l$ , FedHM achieves model compression by reducing the number of FC layer parameters from  $a \times b$  to  $(a + b) \times r$ .

**Convolution Layer Factorization.** We use  $m$  and  $n$  to denote the number of input channels and output channels of the convolution filters, respectively. In a 2D convolution layer, an input with dimension  $(m, H, W)$  is convolved with  $n$  convolution filters of dimension  $(m, k, k)$ , resulting in an  $n$ -channel output feature map. In our notation,  $H, W$  denote the height and width of the input feature map and  $k$  denotes the shape of the convolution filters. The core idea of low-rank factorization is to replace full-rank vanilla convolution layers with factorized versions, reducing both the model size and the computational complexity.

There are multiple ways to adopt factorized convolution layers. In FedHM, we utilize the following strategy. For a 2D convolution layer, the parameter matrix is a 4D tensor  $W$  with dimension  $(m, n, k, k)$ . Instead of directly factorizing the 4D tensor of a convolution layer, we consider factorizing the unrolled 2D matrix. Unrolling the 4D tensor  $W$  results in a 2D matrix with shape  $(mk, nk)$ . By factorizing the matrix  $W \in \mathbb{R}^{mk \times nk}$  via truncated *Singular Value Decomposition* (SVD), we obtain  $U \in \mathbb{R}^{mk \times r}$ ,  $V^T \in \mathbb{R}^{r \times nk}$ , with a specific choice of  $r$ . Reshaping the factorized  $U$  and  $V$  back to 4D tensors gives us  $U \in \mathbb{R}^{m \times r \times k \times 1}$  and  $V^T \in \mathbb{R}^{r \times n \times 1 \times k}$ . Therefore, factorizing a convolution layer with dimension  $(m, n, k, k)$  returns two convolution layers with dimensions  $(m, r, k, 1)$  (a convolution layer with  $r$  convolution filters, each with a size of  $(m \times k \times 1)$ ) and  $(r, n, 1, k)$  (a convolution layer with  $n$  convolution filters, each with a size of  $(r \times 1 \times k)$ ). Note that in this work, we do not consider the tensor decomposition methods [32,34], as they have been observed to result in lower accuracy and greater computational cost compared to SVD [49].

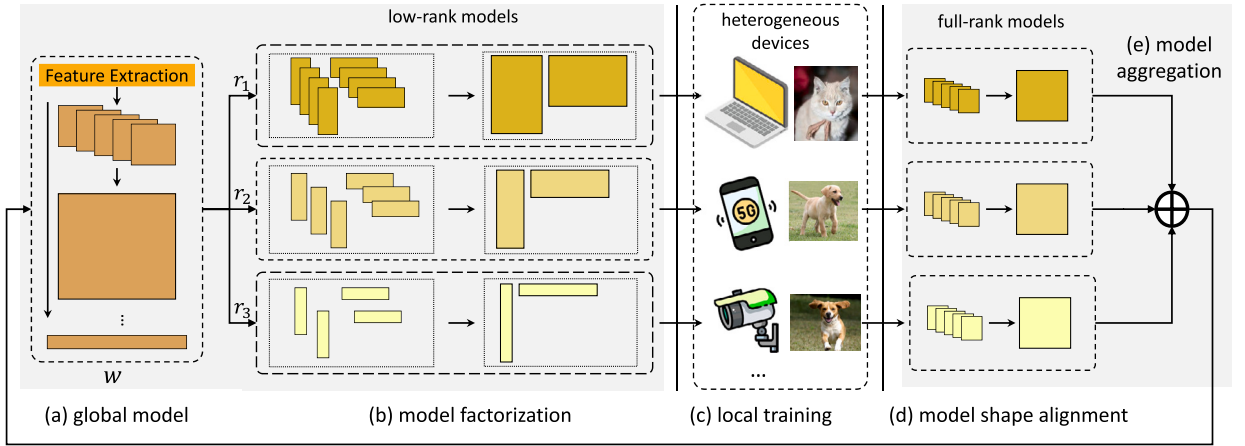


Fig. 2. Overview of FedHM. The global model  $w$  is low-rank factorized to small models and distributed to three clients. The model factorization, model recovery, and model aggregation steps (a), (b), (d), and (e) are on the server side. Only the local training step (c) is run on the client side.

### 3.2. Overview of FedHM

In FedHM, each client participates in FL by training a low-rank factorized neural network with a specified size. This approach allows devices with limited resources to contribute to FL, while also enabling devices with strong computational and communication capabilities to fully utilize their available resources. Prior to the local training stage, the server factorizes the global model  $w$  into various hybrid networks and distributes them to the corresponding participants. Upon receiving the assigned factorized model, clients train it on their private data using low-rank regularization, and subsequently send the updated model back to the server. An overview of FedHM is illustrated in Fig. 2.

FedHM consists of two main components: (1) Factorized training: Factorized training allows clients with heterogeneous hardware resources to train factorized low-rank models of varying sizes. This enables each client to utilize its hardware resources optimally. For example, a smartphone client with 8 GB memory capacity can only train a factorized low-rank ResNet-18 with a compression ratio of  $\frac{1}{8}$ . In contrast, a laptop client with 16 GB memory capacity can afford to train a full-rank ResNet-18. (2) Model shape alignment and model aggregation: After receiving the factorized models, FedHM performs model shape alignment to transform the factorized low-rank models, which have different sizes, back to the same full-rank shape. Specifically, the server transforms factorized models  $U \in \mathbb{R}^{m \times r}$ ,  $V^T \in \mathbb{R}^{r \times n}$  with different values of  $r$  back to  $W \in \mathbb{R}^{m \times n}$ . This ensures that each transformed model has an identical shape and FedHM can aggregate the transformed models using weighted averaging.

### 3.3. Strategies for higher performance

**Hybrid Network Architecture.** Since the low-rank factorized network is an approximation of the original network, approximation errors in the front layers can accumulate and propagate to the later layers, significantly impacting model performance [50]. In CNNs, the number of parameters in the later layers dominates the overall network size. For example, in ResNet-50, approximately 60% of the model parameters are contained in the last three bottleneck residual blocks. To mitigate the negative effects of approximation errors without compromising the compression rate, we propose the hybrid  $[\rho, L, \gamma]$  network architecture, denoted as  $w^\gamma = \{W_1, W_2, \dots, W_\rho, U_{\rho+1}, V_{\rho+1}^T, \dots, U_L, V_L^T\}$ , where  $L$  denotes the total number of layers,  $\rho$  denotes the number of layers that are not factorized,  $\gamma = \text{rank}(U_\ell V_\ell^T) / \text{rank}(W_\ell)$  for all  $\ell \in \{\rho+1, \dots, L\}$ . Note that  $\rho$  and  $\gamma$  are two hyper-parameters that balance the model compression ratio. See Fig. 3.

**Initialization and Regularization of Factorized Layers.** As we factorize the later layers into low-rank forms, directly applying weight decay to these layers is not an optimal choice (i.e., adding  $\frac{\lambda}{2} (\|U\|_2^2 + \|V\|_2^2)$  to the objective, where  $\lambda$  represents the regularization coefficient). Instead, we use *Frobenius Decay* (FD) [49] for regularization of the low-rank layers. The FD penalty on matrices  $U$ ,  $V^T$  is defined as  $\frac{\lambda}{2} \|UV^T\|_F^2$ , regularizing the matrix product. Besides, we initialize the client models with *Spectral Initialization* (SI) during the FL model distribution process. SI leverages the SVD to decompose the server model, ensuring that the low-rank layers are appropriately initialized and regularized.

### 3.4. Model shape alignment

During round  $t$ , after receiving the local update  $w_{p,t}^{\gamma_p}$  from participant  $p$ , where  $\gamma_p$  represents the rank ratio, FedHM performs the model shape alignment operation to recover the full-size model  $w_{p,t}$  as

$$w_{p,t+1} = \{W_1, W_2, \dots, W_\rho, U_{\rho+1} V_{\rho+1}^T, \dots, U_L V_L^T\}, \quad (1)$$

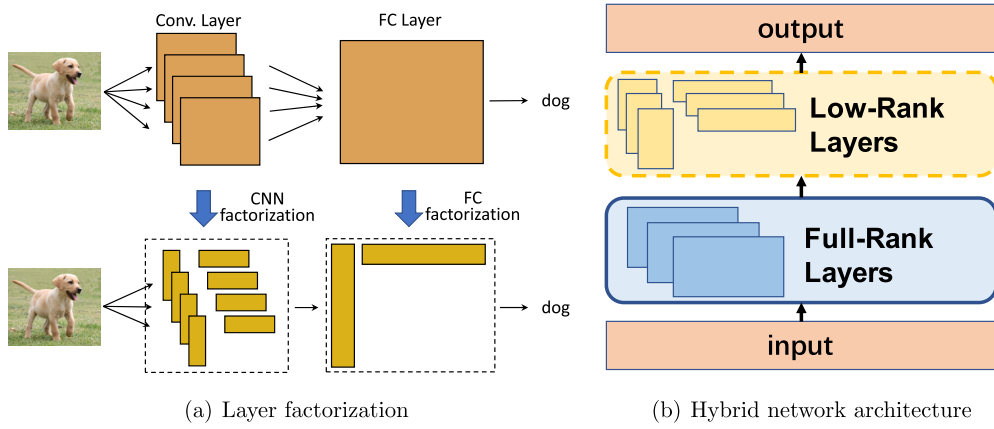


Fig. 3. Factorized training on the client side: (a) Layer factorization: factorization strategies for CNN and FC layers. (b) Hybrid network architecture.

---

**Algorithm 1: FedHM: Federated Learning for Heterogeneous Models.**


---

**Input :** Dataset  $[D_p]_{p=1}^P$  distributed on  $P$  clients, the participation rate  $r_p$ , a set of rank ratios  $\{\gamma_i\}_{i=1}^\Gamma$  with  $\Gamma$  various rank ratios, the number of unfactorized layers  $\rho$ , the number of communication rounds  $T$ .

**Output:** Final global model  $w_T$ .

**ServerExecute:** // Server side

- 1 Initialize a  $L$ -layer network  $w_0 = \{W_1, \dots, W_L\}$
- 2 **for**  $t = 0, \dots, T - 1$  **do**
- 3   **for**  $i = 1, \dots, \Gamma$  **do**
- 4     Factorize the network  $w_t$  to a  $[\rho, L, \gamma_i]$  hybrid network  $w_t^{\gamma_i}$ .
- 5   Randomly select  $[r_p \cdot P]$  clients.
- 6   **for each selected client**  $p$  **do in parallel**
- 7     Based on the computation capability, choose the hybrid network  $w_{p,t}^{\gamma_i}$  from  $\{w_t^{\gamma_i}\}_{i=1}^\Gamma$ .
- 8      $w_{p,t+1}^{\gamma_i} \leftarrow \text{LocalUpdate}(w_{p,t}^{\gamma_i})$ .
- 9     Perform model shape alignment on the hybrid network  $w_{p,t+1}^{\gamma_i}$  to get  $w_{p,t+1}$  as in Eq. (1).
- 10   Aggregate the local updates from clients to get a new network weight  $w_{t+1}$ , i.e.  $w_T$ , as Eq. (2).
- 11 **return**  $w_T$

---

To avoid cluttered notation on the right side of Eq. (1), we drop the subscripts for  $p$  and  $t$ . As an example, consider the low-rank matrices  $U_l \in \mathbb{R}^{b \times r}$  and  $V_l \in \mathbb{R}^{a \times r}$  of the  $l$ -th layer of the model, the full-rank matrix  $W \in \mathbb{R}^{a \times b}$  can be obtained through the  $U_l V_l^\top$  operation, as shown in Equation (1). In this way, low-rank matrices of different ranks are restored to full-rank matrices of the same size.

### 3.5. Model aggregation

As the number of parameters in each model represents the knowledge it possesses, we use a weighted aggregation scheme to derive the global model from the local updates, defined as

$$w_{t+1} = \sum_{p=1}^P \alpha_p w_{p,t+1}, \quad \alpha_p = \frac{\exp\left(\frac{\gamma_p}{\tau}\right)}{\sum_{p=1}^P \exp\left(\frac{\gamma_p}{\tau}\right)} \quad (2)$$

where  $P$  is the number of active clients,  $\alpha_p$  denotes the weight of  $w_{p,t+1}$ , and  $\tau > 0$  denotes the softmax temperature. In the experiments,  $\tau$  is set to 1 by default. The proposed FedHM is described in Algorithm 1.

### 3.6. Discussion on privacy and security

Privacy and security are critical considerations in FL. Traditional FL, which involves the exchange of the complete model between clients and the server, has proven to be a reliable approach [51]. In the context of our proposed FedHM, the use of low-rank matrices ensures comparable privacy and security to full-rank matrices, as low-rank representations retain the same expressive power. Furthermore, since only the low-rank layers of the hybrid network model are transmitted, the risk of private data exposure through membership inference or reconstruction attacks [52] is significantly mitigated. Even during an attack, adversaries would lack sufficient information to fully reconstruct sensitive data. Consequently, our proposed FedHM not only maintains robust privacy and security but also offers enhanced reliability compared to non-low-rank methods.

**Table 1**

Comparison of parameters and computational complexity for full-rank/low-rank FC and Conv. layers. Conv. is short for Convolution; Params is short for Parameters; Comp. is short for Computational.

Layer	# Params.	Comp. Complexity
Vanilla FC	$mn$	$\mathcal{O}(mn)$
Factorized FC	$r(m+n)$	$\mathcal{O}(r(m+n))$
Vanilla Conv.	$mnk^2$	$\mathcal{O}(mnk^2HW)$
Factorized Conv.	$rk(m+n)$	$rkHW(m+n)$
Pufferfish	$r(mk^2+n)$	$rHW(mk^2+n)$

#### 4. Theoretical analysis

In this section, we first conduct a theoretical analysis of the computational complexity and model size of FedHM and then provide proof of the convergence of FedHM.

##### 4.1. Computational complexity and model size

We summarize the computational complexity and the number of parameters in the vanilla/low-rank FC layers and convolution layers in Table 1. Low-rank factorization reduces the dimension of FC layers from  $(m, n)$  to  $(m, r), (r, n)$ . In terms of computational complexity,  $xW$  requires  $\mathcal{O}(mn)$  and  $xUV^T$  requires  $\mathcal{O}(mr + rn) = \mathcal{O}(r(m+n))$ . For a convolution layer, the unfactorized layer has dimension  $(m, n, k, k)$ , while the factorized low-rank layer has dimensions  $(m, r, k, 1), (r, n, 1, k)$  as previously discussed. Therefore, factorized convolution reduces the total number of parameters from  $mnk^2$  to  $rmk + rnk$ . Regarding computational complexity, an unfactorized convolution layer requires  $\mathcal{O}(mnk^2HW)$  operations, as  $n$  filters with size  $m \times k \times k$  need to convolve over  $H \times W$  pixels (controlled by a stride size). Using the same analysis, the factorized low-rank convolution layer needs  $\mathcal{O}(mrkHW + nrkHW)$  operations to convolve an input with  $H \times W$  pixels. We also compare our convolution layer factorization strategy against another popular factorization strategy used in Pufferfish [50]. For both model size and computational complexity, our factorization strategy balances the  $k$  term between the input dimension  $m$  and output dimension  $n$  while the strategy used by Pufferfish places all  $k^2$  terms in the input dimension  $m$ . Thus, unless  $m$  is significantly smaller than  $n$ , our strategy usually leads to smaller model sizes and lower computational complexity. Notably, the reduction in the number of parameters improves both computation and communication efficiencies.

##### 4.2. Convergence analysis

We provide a convergence result for Algorithm 1. We consider the minimization of the loss function  $f$ , which can be formulated as:  $\min_w f(w) = \frac{1}{P} \sum_{p=1}^P f_p(w)$ . Here,  $P$  is the number of participants, each participant holds local data  $\mathcal{D}_p$  which contains sample  $\zeta_p$ , and each  $f_p := \mathbb{E}_{\zeta_p \sim \mathcal{D}_p} [F_p(w; \zeta_p)]$  is a smooth non-convex function where each participant has their local dataset  $\mathcal{D}_p$ .

**Assumption 1.** Each function  $f_p(w)$  is continuously differentiable, the gradients of  $f_p(w)$  are Lipschitz with modulus  $L_{\nabla f}$ , and the function  $f(w)$  is bounded below by  $f_{\text{low}} \in \mathbb{R}$ . For each participant  $p$  and iteration  $t$ , the stochastic gradients satisfy  $\mathbb{E}_{p,t}[\nabla F_p(w_{p,t}; \zeta_{p,t})] = \nabla f_p(w_{p,t})$ . Additionally, there exist constants  $M_f > 0$  and  $\kappa_f > 0$  satisfy

$$\begin{aligned} \mathbb{E}_{\zeta_p \sim \mathcal{D}_p} [\|\nabla F_p(w; \zeta_p) - \nabla f_p(w)\|^2] &\leq M_f, \quad \forall w, \forall p \\ \mathbb{E}_{\zeta_p \sim \mathcal{D}_p} [\|\nabla F_p(w; \zeta_p)\|^2] &\leq \kappa_f, \quad \forall w, \forall p \end{aligned} \quad (3)$$

where  $\|\cdot\|$  denotes the Euclidean norm of a vector,  $F_p(*)$  denotes output of participant  $p$ 's local model.

**Assumption 2.** There exists a constant  $\kappa_{UV} > 0$ , at every iteration  $t$ , we have

$$\|U_{p,t}^\ell\|_F \leq \kappa_{UV}, \quad \|V_{p,t}^\ell\|_F \leq \kappa_{UV}, \quad \forall p, \forall \ell \in \{\rho+1, \dots, L\} \quad (4)$$

where  $\|\cdot\|_F$  denotes the Frobenius norm of a matrix,  $L$  represents the number of layers in the network.

**Assumption 3.** There exist constants  $\kappa_{\sigma_1}$  and  $\kappa_{\sigma_2}$ , at every iteration  $t$  such that  $t \bmod E = 0$ , we have

$$\mathbb{E} \left[ \left\| \left( \frac{1}{P} \sum_{p=1}^P \sqrt{\Sigma_{r(p,\ell),t}^\ell} \right) \left( \frac{1}{P} \sum_{p=1}^P \sqrt{\Sigma_{r(p,\ell),t}^\ell} \right) - \Sigma_t^\ell \right\|_F^2 \right] \leq \kappa_{\sigma_1} \quad (5)$$

$$\text{and } \mathbb{E} \left[ \left\| \frac{1}{P} \sum_{i=1}^P \sqrt{\Sigma_{r(i,\ell),t}^\ell} - \sqrt{\Sigma_{r(p,\ell),t}^\ell} \right\|_F^2 \right] \leq \kappa_{\sigma_2}$$

where  $E$  is defined as the number of local epochs,  $\Sigma_{r(p,\ell),t}^\ell = \text{diag}(\sigma_1, \dots, \sigma_{r(p,\ell)}, 0, \dots, 0)$ , denotes the eigenvalues of the matrix after low-rank factorization,  $r(p, \ell)$  denotes the rank of  $U_{p,t}^\ell$  and  $V_{p,t}^\ell$  for all  $\ell \in \{\rho + 1, \dots, L\}$ .

Assumption 1 is a common assumption in the literature for non-convex parallel restarted SGD [53]. Assumption 2 is specific to the low-rank factorization strategy of Algorithm 1. Assumption 3 bounds the difference between the server model and the low-rank models during each aggregation step. Due to space limitations, detailed theoretical proof can be found in Appendix A.

**Theorem 1.** Let Assumptions 1, 2, and 3 hold, we have

$$\frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}[\|\nabla g(\hat{h}_t)\|^2] \leq \frac{2}{T\eta} (f(w_0) - f_{\text{low}}) + \frac{2L_f(L - \rho)\kappa_{\sigma_1}}{E\eta}$$

$$+ 6\eta\kappa_g E \left( \eta L_{\nabla g}^2 E + 2L_f \kappa_{UV}^2 \right) + \frac{\eta L_{\nabla g} M_g}{P} + (6(L - \rho)\kappa_{\sigma_2}) \left( L_{\nabla g}^2 + \frac{2L_f \kappa_{UV}^2}{E\eta} \right)$$

where  $\eta$  is the learning rate,  $L_f$  is the Lipschitz constant of the function  $f$ , and  $g(\hat{h}_t)$ s represents the loss function of the global model and the local model.

## 5. Extensions for pre-trained models

In this section, we first introduce the LoRA fine-tuning method for Transformer-based models, and then introduce our proposed FedLoRA for model heterogeneous federated learning scenarios.

### 5.1. Low-rank adaptation for transformers

The encoder transformer block [54–56] contains multi-head self-attention (MSA) layers and Multilayer Perceptron (MLP). An MSA layer with  $H$  heads takes an input sequence  $x \in \mathbb{R}^{n \times d}$  with length  $n$  and hidden dimension  $d$ , and projects the input to key ( $K_h$ ), query ( $Q_h$ ) and value ( $V_h$ ), then outputs a weighted sum based on the attention scores  $A_h$ :

$$Q_h = xW_h^Q, K_h = xW_h^K, V_h = xW_h^V, \quad (6)$$

$$A_h = \text{softmax}\left(\frac{Q_h K_h^\top}{\sqrt{d/H}}\right) V_h, \quad (7)$$

$$MSA(x) = [A_1 V_h; A_2 V_h; \dots; A_H V_h] W_O, \quad (8)$$

where the parameter matrices  $W_h^Q, W_h^K, W_h^V \in \mathbb{R}^{d \times d/H}$ , and  $W_O \in \mathbb{R}^{d \times d}$ .

Fine-tuning pre-trained transformers always update all the parameters of the original model. Therefore, adapting to different downstream tasks incurs considerable storage and computation overhead [55,56]. To reduce the number of trainable parameters, we fix all pre-trained weights and add low-rank bypass matrices for attention weight matrices  $W_h^Q, W_h^K, W_h^V$  and  $W_O$ . Eq. (6) and Eq. (8) are modified as:

$$Q_h = x(W_h + B_h C_h^\top)^Q, K_h = x(W_h + B_h C_h^\top)^K, V_h = x(W_h + B_h C_h^\top)^V, \quad (9)$$

$$MSA(x) = [A_1 V_h; A_2 V_h; \dots; A_H V_h] (W_O + B_O C_O^\top), \quad (10)$$

where  $W_h$  and  $W_O$  represent fixed pre-trained weights,  $B_h \in \mathbb{R}^{d \times r/H}$ ,  $C_h \in \mathbb{R}^{r/H \times d/H}$ ,  $B_O \in \mathbb{R}^{d \times r}$ ,  $C_O \in \mathbb{R}^{r \times d}$ , and  $r \ll d$ . As illustrated in Fig. 4(a), we fix  $W_h$  and  $W_O$ , and set  $B_h, B_O, C_h, C_O$  as trainable parameters with low-rank adaptation.

### 5.2. Low-rank adaptation in federated learning

Pre-trained transformers have been proven to be robust to data heterogeneity in FL [57]. Despite this, the high computational and communication costs associated with transformers hinder their practical implementation in FL. To address this issue, we propose a compression strategy named FedLoRA, wherein each client stores and fixes the pre-trained weights while only transmitting the



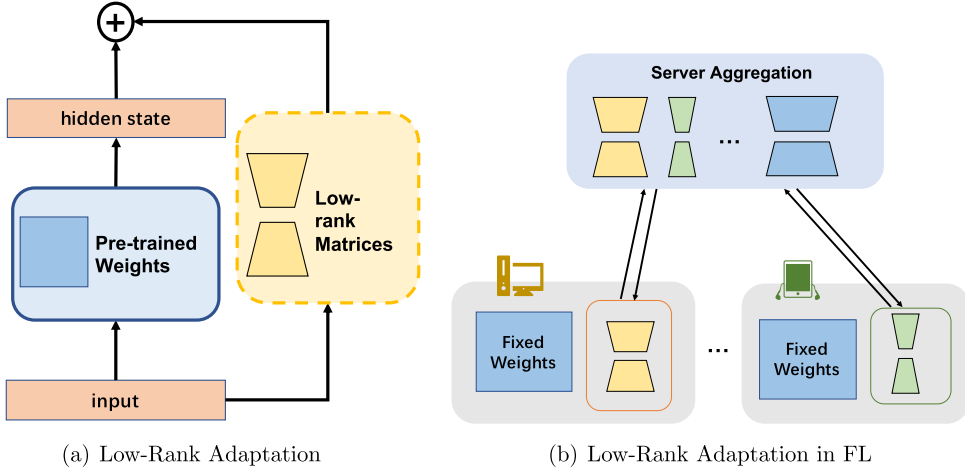


Fig. 4. Overview of FedLoRA. The low-rank matrices are trainable parameters, and they are transmitted to the server.

**Table 2**  
Detailed local model parameter (M).  $g_1, g_2, g_3, g_4$  denote 4 complexity levels.

Model	Method	$g_1$	$g_2$	$g_3$	$g_4$
ResNet-18	HeteroFL	11.17	4.29	2.80	1.37
	FedRolex	12.33	4.12	2.75	1.37
	Split-Mix	11.17	4.16	2.21	1.24
	FedHM	11.17	4.16	2.21	1.24
ResNet-34	HeteroFL	21.33	8.77	5.35	3.42
	FedRolex	23.97	10.22	6.18	3.42
	Split-Mix	21.33	8.40	4.99	3.27
	FedHM	21.33	8.40	4.99	3.27

trained low-rank matrices to the server in each communication round. Fig. 4(b) illustrates how FedLoRA significantly reduces the number of trainable parameters for fine-tuning the pre-trained transformers. Furthermore, we apply the same aggregation strategy as FedHM for the low-rank matrices in FedLoRA.

## 6. Experiments

In this section, we first describe the experimental setup, then make a detailed comparison between FedHM<sup>1</sup> and the state-of-the-art FL baseline methods in terms of accuracy, convergence speed, communication and computation cost, and finally provide the experimental performance of FedLoRA.

### 6.1. Experimental setup

**Experimental Platform.** Our implementation is based on PyTorch and utilizes GLOO as the communication backend. All the experiments are conducted on a 256 GB server with 4 Tesla V100 GPUs running in parallel.

**Datasets and Model Setting.** To validate the effectiveness of our proposed approach, we conduct experiments on CIFAR-10 [58] and CIFAR-100 [58] datasets. The CIFAR-10 dataset comprises 60,000 images with a resolution of  $32 \times 32$ , encompassing 10 classes, each containing 6,000 images. Similarly, CIFAR-100 is an image dataset comprising 100 classes, with 600 images per class. Based on the complexity of the datasets, we conduct experiments using ResNet-18 for CIFAR-10 and ResNet-34 for CIFAR-100. In addition, to evaluate FedLoRA, we utilize the *Vision Transformer* (ViT)-Small [55] as the pre-trained model. To adapt to the FL setting, we set the number of clients to 100 and sample 10% of the clients during each round for training, with 10 local epochs. Following [20], we incorporate static batch normalization (sBN) and masking cross-entropy methods into all the methods in the experiments. The details of the model architecture are provided in Appendix B.

**Baselines.** We compare FedHM with the following baselines: (1) KD-based methods: FedDF, DS-FL, Fed-ET. (2) PT-based methods: HeteroFL, Split-Mix, FedRolex. To ensure a fair comparison, FedHM and all the PT-based baselines are trained under the same hyperparameters.

<sup>1</sup> The source code is available at <https://github.com/CGCL-codes/FedHM>.



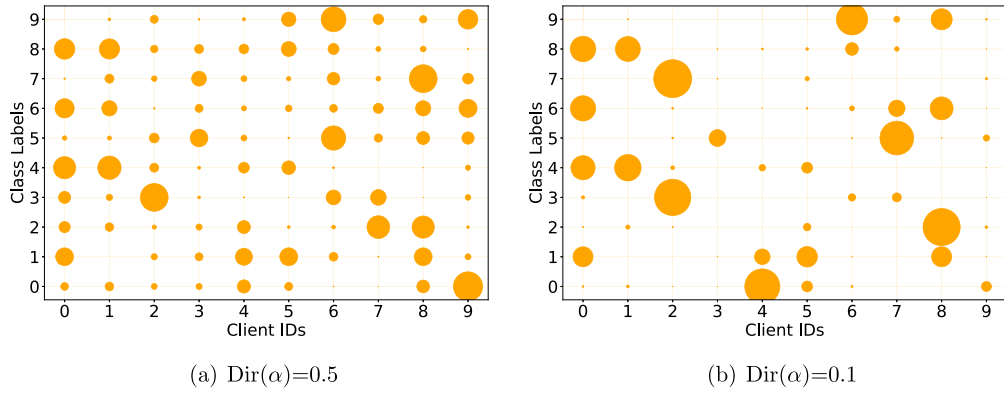


Fig. 5. Illustration of data heterogeneity among 10 clients on CIFAR-10, where the x-axis indicates client IDs, the y-axis indicates class labels, and the size of dots indicates # of samples per class allocated to each client.

**Table 3**  
Detailed hyper-parameters.

Dataset	CIFAR-10	CIFAR-100
Number of Clients	100	
Client Participation Rate	0.1	
Local Epoch	10	
Batch Size	64	
Optimizer	SGD+0.9Momentum	
Weight decay	5e-4	
Learning Rate	0.01	
Local Model	ResNet-18	ResNet-34
Number of Communication Rounds	500	1000

**Table 4**  
Top-1 global model accuracy comparison among FedHM, PT and KD-based model-heterogeneous FL methods. Similar to the FedRolex paper, the results of KD-based methods are obtained from [17].

Method		IID ( $\alpha = 0.5$ )		Non-IID ( $\alpha = 0.1$ )	
		CIFAR-10	CIFAR-100	CIFAR-10	CIFAR-100
KD-based	FedDF [15]	76.55±0.32	37.87±0.31	73.81±0.42	31.87±0.46
	DS-FL [16]	68.44±0.47	33.56±0.55	65.27±0.53	29.12±0.51
	Fed-ET [17]	81.13±0.28	41.58±0.36	78.66±0.31	35.78±0.45
PT-based	HeteroFL [20]	79.65±0.25	55.68±0.36	75.47±0.32	52.84±0.61
	Split-Mix [21]	81.16±1.26	56.49±0.47	77.41±0.46	53.82±0.85
	FedRolex [22]	83.99±0.97	57.85±0.62	78.83±1.03	54.38±0.75
FedHM		<b>88.84±0.43</b>	<b>59.15±0.21</b>	<b>84.46±0.57</b>	<b>57.98±0.45</b>

**System Heterogeneity.** The system heterogeneity in our experiments is pre-defined as different levels of computational complexity. We set up models with 4 complexity levels, denoted as  $g_1, g_2, g_3$ , and  $g_4$ . The models are ranked in decreasing order of size, where  $g_1$  corresponds to the original uncompressed model. For HeteroFL and FedRolex, we adjust the compression ratio as  $\{1.0, 0.64, 0.5, 0.35\}$  to vary the model size, corresponding to complexity levels  $g_1, g_2, g_3, g_4$ . For Split-Mix, we control the size of the model by adjusting the number of base models, denoted as  $k$ . We set  $k = \{9, 3, 2, 1\}$  for  $g_1, g_2, g_3, g_4$ . Due to differences in compression strategies, it is challenging to set the heterogeneous model parameters of our proposed FedHM exactly the same as those of the baselines. To compensate, we compress the local models of FedHM into fewer parameters. Specifically, we set the rank ratio  $\gamma = \{1, 2, 4, 8\}$ , which corresponds to  $g_1, g_2, g_3, g_4$ , respectively. Additionally, as mentioned in Section 3.3, we use  $\rho$  to denote the number of layers that are not factorized. We set  $\rho = 2$  on ResNet-18 and  $\rho = 15$  on ResNet-34. Table 2 shows more detailed information about the parameters of the local model.

**Data Heterogeneity.** Following previous work [59], we sample disjoint Non-IID client data using the Dirichlet distribution  $\text{Dir}(\alpha)$ , where  $\alpha$  denotes the concentration parameter. When allocating data to each client, a distribution vector can be sampled from the Dirichlet distribution based on  $\alpha$ , and then the global training data can be assigned to different clients according to this distribution vector. A smaller  $\alpha$  indicates a higher degree of data heterogeneity. Following the settings of FedRolex [22], we set  $\alpha = 0.5$  for IID scenarios and  $\alpha = 0.1$  for Non-IID scenarios. Fig. 5 presents how data samples are scattered among 10 clients on the CIFAR-10 dataset.

**Configuration.** We run experiments in a scenario consisting of 100 clients and 1 server. The client sampling rate for each communication round is set to 0.1, which means that 10 clients will participate in the training in each round. The batch size of the client's

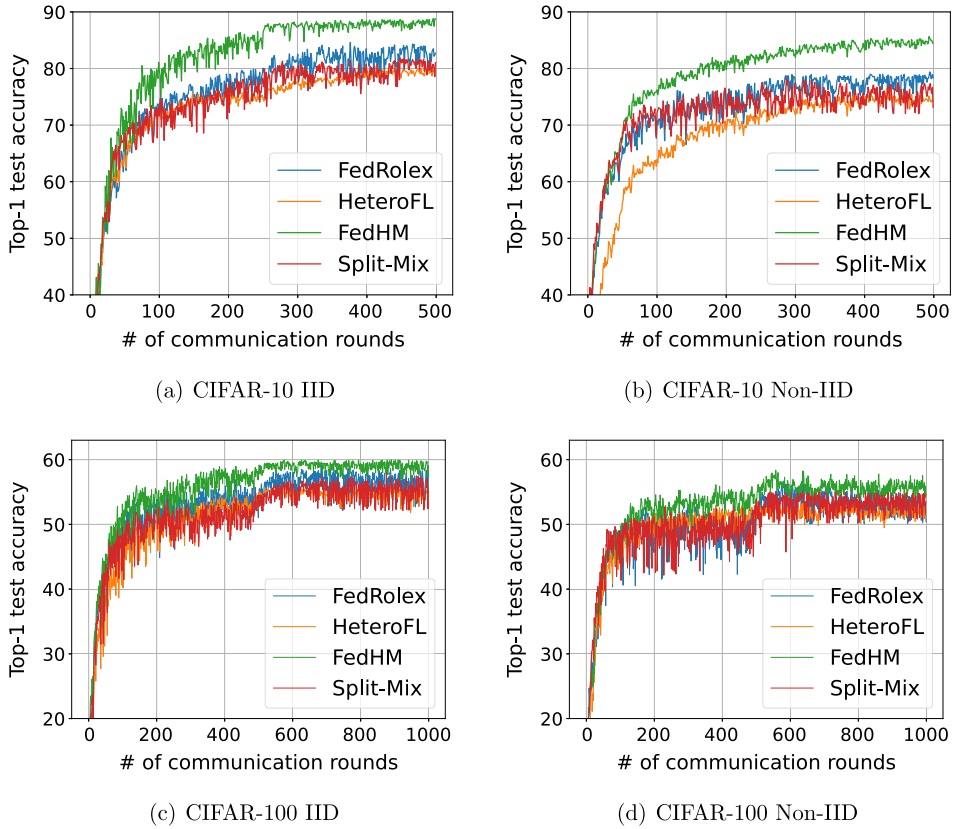


Fig. 6. Convergence speed evaluation of HeteroFL, Split-Mix, FedRolex and FedHM.

local training input data is 64, and the SGD optimizer with a momentum of 0.9 is used for 10 local training epochs. The learning rate is 0.1 and the weight decay coefficient is  $5e-4$ . For the CIFAR10 dataset, the full-rank model used is ResNet-18 and trained for 500 rounds; for the CIFAR100 dataset, the full-rank model is set to ResNet-34 and trained for 1000 rounds. Table 3 presents the detailed hyper-parameters in our experiments.

**Experiment Settings for FedLoRA.** Due to the strong robustness of ViT, we conduct experiments in highly heterogeneous data scenarios, with  $\alpha$  set to 0.01. The experiments are trained on the CIFAR-10 dataset. To align the input dimensions with ViT-Small, we resize the CIFAR-10 images to  $224 \times 224$  pixels. We choose the dimension  $r$  from a set of  $\{2, 4, 8, 16\}$ , resulting in compressed models with sizes of 0.06M, 0.11M, 0.22M, and 0.44M, respectively. We set 40 communication rounds for the whole training process. In each communication round, we choose Adam as the optimizer with a learning rate of 0.002, and each client is trained for 1 local epoch.

## 6.2. Performance evaluation

Table 4 demonstrates that FedHM outperforms the baselines under both IID and Non-IID conditions. Specifically, on CIFAR-10, FedHM surpasses HeteroFL by 9.19% and 8.99% under IID and Non-IID settings, respectively; surpasses Split-Mix by 7.68% and 7.05% under IID and Non-IID settings respectively; surpasses FedRolex by 4.85% and 5.63% under IID and Non-IID settings respectively. On CIFAR-100, FedHM surpasses HeteroFL by 3.47% and 5.14% under the IID and Non-IID settings; surpasses Split-Mix by 3.01% and 4.16% under the IID and Non-IID settings; surpasses FedRolex by 1.30% and 3.60% under the IID and Non-IID settings.

## 6.3. Efficiency analysis

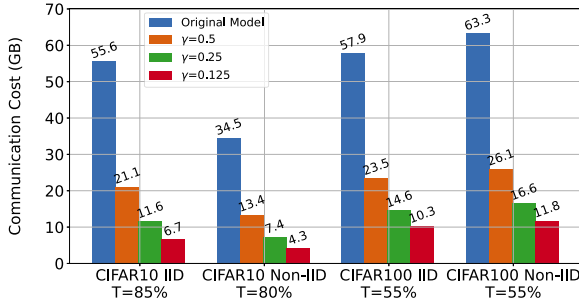
**Convergence Speed under Heterogeneous Setting.** Fig. 6 illustrates that FedHM achieves faster convergence compared to baselines under IID data partitioning, thus demonstrating the benefits of FedHM's factorized training and aggregation strategy.

**Communication and Computation Cost.** Fig. 7 shows the communication and computation costs required to achieve the target accuracy for factorized low-rank models. We compute the communication cost as  $2 \times (\#participants) \times (\#model\ parameters) \times (\#round)$ , and the computation cost is calculated as  $(MACs) \times (\#epochs) \times (\#samples) \times (\#round)$ , where  $\#MACs$  denotes the number of multiplication-and-addition operations. As observed in Fig. 7(a), reducing the model size gradually decreases the communication cost required to achieve the target accuracy. For instance, compared to the original model, the low-rank model  $g_4$  reduces the communication cost  $8.3 \times / 8.0 \times$  and  $5.6 \times / 5.8 \times$ , under the CIFAR10 IID/Non-IID and CIFAR100 IID/Non-IID settings. Fig. 7(b) demonstrates that the original model requires  $4.0 \times / 3.8 \times$  and  $1.7 \times / 1.6 \times$  higher computation cost than the low-rank model  $g_4$  to achieve a target

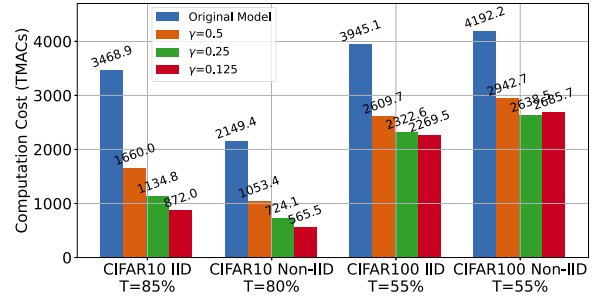
**Table 5**

Comparison of FedLoRA, Adapter, Prefix-Tuning, and SLoRA on the CIFAR-10 dataset. There are 80 clients in total, with a client participation rate of 10%. M1, M2, M3, and M4 represent the local models with 0.06M, 0.11M, 0.22M, and 0.44M trainable parameters, respectively.

Method	M1	M2	M3	M4
Adapter [47]	72.03	75.58	76.79	77.01
Prompt-Tuning [60]	72.47	75.90	77.03	77.24
Prefix-Tuning [48]	73.15	76.13	77.45	77.92
SLoRA [61]	78.96	80.06	80.36	80.55
FedLoRA	79.35	80.12	80.29	80.50



(a) Communication cost of FedHM in FL training.



(b) Computation cost of FedHM on each client.

**Fig. 7.** Communication cost and computation cost evaluation to reach target Top-1 test accuracy under different rank ratio  $\gamma$ .  $T$  denotes the target accuracy.  $g_1$  corresponds to the original model,  $g_2, g_3, g_4$  correspond to  $\gamma = 0.5, \gamma = 0.25, \gamma = 0.125$  respectively.

accuracy, under the CIFAR10 IID/Non-IID and CIFAR100 IID/Non-IID settings, respectively. The experimental results validate the effectiveness of local factorized training and indicate that factorized training further enhances the performance of client models in heterogeneous settings.

#### 6.4. Experiments on FedLoRA

We evaluate the proposed FedLoRA from two aspects: model performance and training efficiency.

##### 6.4.1. High performance

FedTune [45] and FedPETuning [46] explore federated fine-tuning techniques, such as Adapter [47], Prompt-Tuning [60], Prefix-Tuning [48], and LoRA [24], under the assumption of model homogeneity. SLoRA [61], on the other hand, proposes using unified parameters to initialize the low-rank matrices of different clients to address the non-IID data partitioning problem in FL. To ensure a fair comparison, we set the same number of trainable parameters across all methods. Specifically, the hidden layer dimensions of Adapter are set to  $\{8, 16, 32, 64\}$ ; the hidden layer dimensions after re-parameterization for Prefix-Tuning are set to  $\{4, 8, 16, 32\}$ ; the prefix embedding lengths for Prompt-Tuning are set to  $\{16, 32, 64, 128\}$ . Since the aggregation of trainable parameters across heterogeneous models is not feasible for Adapter and Prefix-Tuning, we only aggregated homogeneous models for these methods.

The comparative experimental results for FedLoRA, Adapter, Prefix-Tuning, and SLoRA are presented in Table 5. The results demonstrate that FedLoRA performs significantly better than both Adapter and Prefix-Tuning, indicating that FedLoRA fine-tuning is more efficient with the same number of trainable parameters. Furthermore, FedLoRA achieves comparable performance to SLoRA, suggesting that the use of the same random initialization parameters across clients has minimal impact on the final training results.

##### 6.4.2. High efficiency

Fig. 8 illustrates the performance of pre-trained ViT-Small under an extreme Non-IID setting ( $\alpha = 0.01$ ). In the experiments, full fine-tuning (FT) training refers to fine-tuning all parameters of the original model (21.70 MB), FedLoRA represents deploying low-rank matrices of the same size across all clients (0.06M, 0.11M, 0.22M, and 0.44M respectively), FedHM indicates that each client deploys heterogeneous low-rank matrices with sizes of 0.06M, 0.11M, 0.22M, and 0.44M. As depicted in Fig. 8, FedLoRA outperforms FT with a reduction of up to 50 $\times$  of trainable parameters, significantly improving communication and computational efficiency. However, when the low-rank bypass is over-compressed, i.e., the model parameters are reduced below 0.1M, its performance drops significantly.

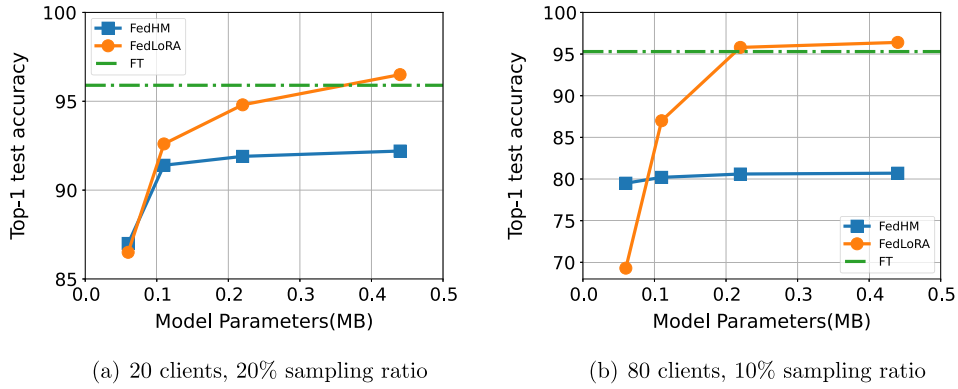


Fig. 8. Evaluation of FedLoRA and FedHM on CIFAR-10 under an extreme Non-IID setting ( $\alpha = 0.01$ ).

Table 6

Top-1 global model accuracy comparison between FedHM and PT-based model-heterogeneous FL methods on Tiny-ImageNet.

Data Partition	HeteroFL	Split-Mix	FedRolex	FedHM
IID	47.12 $\pm$ 0.52	47.25 $\pm$ 0.86	48.32 $\pm$ 0.74	<b>49.47<math>\pm</math>0.59</b>
Non-IID	35.53 $\pm$ 0.54	36.29 $\pm$ 0.78	37.45 $\pm$ 0.83	<b>38.61<math>\pm</math>0.36</b>

### 6.5. Experiments on the tiny-ImageNet dataset

To study the performance of FedHM on more complex tasks, Table 6 presents the results of FedHM and baselines on the Tiny-ImageNet dataset. The Tiny-ImageNet dataset<sup>2</sup> consists of 100,000 downsized color images of size  $64 \times 64$ , distributed across 200 classes, with 500 images per class. We utilize the ResNet-34 model for training, keeping the hyper-parameters consistent with those used with the CIFAR-100 dataset. FedHM exhibits superior performance compared to baselines under both IID and Non-IID conditions.

## 7. Conclusion

In this paper, we propose a novel model heterogeneous FL framework FedHM, which aims to reduce both the training and communication overhead on devices. To the best of our knowledge, we are the first to provide a convergence guarantee for the low-rank factorization-based heterogeneous FL framework. By incorporating low-rank decomposition approaches, FedHM improves model performance in various heterogeneous settings. Additionally, we propose FedLoRA, a method designed specifically for fine-tuning pre-trained transformer models efficiently. Comprehensive experiments and analyses verify the effectiveness of our proposed approach compared to state-of-the-art baselines. One limitation of FedHM lies in the empirical setting of the number of unfactorized layers  $\rho$ . The adaptive determination of  $\rho$  is left for future work.

### CRediT authorship contribution statement

**Dezhong Yao:** Writing – review & editing, Writing – original draft, Supervision, Project administration, Funding acquisition, Formal analysis, Conceptualization. **Wanning Pan:** Writing – original draft, Methodology, Formal analysis. **Yuxin Shi:** Writing – review & editing, Software. **Michael J. O'Neill:** Writing – review & editing, Formal analysis. **Yutong Dai:** Writing – review & editing, Formal analysis. **Yao Wan:** Writing – review & editing. **Peilin Zhao:** Writing – review & editing. **Hai Jin:** Project administration, Funding acquisition. **Lichao Sun:** Writing – review & editing, Methodology.

### Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Dezhong Yao reports financial support was provided by National Key Research and Development Program of China. Dezhong Yao reports financial support was provided by National Natural Science Foundation of China. Dezhong Yao reports equipment, drugs, or supplies was provided by National Supercomputing Center in Zhengzhou. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

<sup>2</sup> <https://www.kaggle.com/c/tiny-imagenet>.

**Table A.7**  
Symbols related to convergence analysis.

Symbol	Explanation
$P$	the number of participants in FL training
$E$	local training epochs
$t$	total number of iterations
$w$	model weight
$W^i$	the $i$ -th layer's weight of the full rank model
$U_i, V_i$	low-rank matrix obtained by low-rank factorization of $W_i$
$h$	local hybrid network with low-rank factorized layers
$y$	global model (full rank model)
$L$	the total number of layers in the network
$\rho$	the index of the first low-rank factorization network layer
$\ \cdot\ $	the Euclidean norm of a vector
$\ \cdot\ _F$	the Frobenius norm of a matrix
$D_p$	participant $p$ 's local data
$\zeta$	an element in the dataset $D_p$ of participant $p$
$g_p(h), f_p(w)$	loss function of participant $p$
$F_p(*)$	output of participant $p$ 's local model

## Acknowledgements

This work is supported by the National Key Research and Development Program of China under Grant No.2022YFB4502001 and the National Natural Science Foundation of China under Grant No.62072204. The computation is completed in the HPC Platform of Huazhong University of Science and Technology.

## Appendix A. Convergence analysis

### A.1. Notation and definitions

Table A.7 presents the symbols related to convergence analysis. We utilize  $w$  to denote variables in the weight space, which can be expressed as  $w = \{W^0, \dots, W^L\}$ , where  $W^\ell$  denotes the weight matrix at layer  $\ell$ . We utilize  $h$  to denote variables in the factorized space, which can be expressed as  $h = \{W^0, \dots, W^\rho, U^{\rho+1}, V^{\rho+1}, \dots, U^\ell, V^\ell\}$ . Occasionally, we convert between these representations. To this end, we define

$$\text{inv}(h) = \{W^0, \dots, W^\rho, U^{\rho+1}(V^{\rho+1})^\top, \dots, U^L(V^L)^\top\}. \quad (\text{A.1})$$

where we assume that the quantities  $w$  and  $h$  have been vectorized for ease of notation unless we work specifically with a particular layer  $W^\ell$  or  $U^\ell, V^\ell$ , in which case we work with the matrix representation. To enhance readability, we generally adopt the convention of using lowercase letters for vectors and uppercase letters for matrices, except in rare cases when explicitly noted otherwise. We use  $\|\cdot\|$  to denote the Euclidean norm of a vector and  $\|\cdot\|_F$  to denote the Frobenius norm of a matrix. It should be noted that, for any matrix  $A$ ,  $\|\text{vec}(A)\| = \|A\|_F$ , where  $\text{vec}(A)$  represents the vectorization of  $A$ .

Now, we recall the definition of  $f$ :

$$\min_w f(w) = \frac{1}{P} \sum_{p=1}^P f_p(w), \quad (\text{A.2})$$

where  $P$  is the number of participants and each  $f_p(w) := \mathbb{E}_{\zeta_p \sim D_p} [F_p(w; \zeta_p)]$  is a smooth non-convex function where  $D_p$  can vary per participant. Then, we define the factorized loss function,  $g$ , as

$$\min_h g(h) = \frac{1}{P} \sum_{p=1}^P g_p(h), \quad (\text{A.3})$$

such that, for any  $h$  and  $w$ , we have  $\text{inv}(h) = w$  and  $g_p(h) = f_p(w)$  holds for all  $p$ .

In order to ease our analysis, we provide a slightly modified version of Algorithm 1. Fundamentally, the only differences are:

- A modified definition of communication rounds and local epoch, to ease the notation for the iteration counter in the analysis. We remark that  $E$  represents the communication frequency, i.e., clients upload their “local models” every  $E$  steps to the server to produce a global model. And  $t$  represents the iteration counter, i.e., the total number of epochs since the first communication round.
- We prove a convergence guarantee for FedHM in the model-heterogeneous FL setting.
- The use of stochastic gradient steps, as opposed to a generic training algorithm.

We adopt the generic notation  $y$  to denote the global models and  $h$  to denote the local models (dependence on the iteration or clients is made clear via subscripts).

Now, we present some fundamental properties of learning with factorized layers [50]. For all iterations  $t$ , participants  $p$ , and layers  $\ell \in \{\rho + 1, \dots, L\}$ , let  $U_{p,t}^\ell \in \mathbb{R}^{m \times r}$  and  $V_{p,t}^\ell \in \mathbb{R}^{n \times r}$  denote the factors of layer  $\ell$  for participant  $p$  at time  $t$ . Then, the layer-wise gradient  $\nabla g_p^\ell(h_{p,t})$  and stochastic gradient  $\nabla G_p^\ell(h_{p,t}; \zeta_{p,t})$  satisfy

$$\begin{aligned} \nabla g_p^\ell(h_{p,t}) &= \begin{bmatrix} \nabla f_p^\ell(w_{p,t}) V_{p,t}^\ell \\ \nabla f_p^\ell(w_{p,t})^\top U_{p,t}^\ell \end{bmatrix} \text{ and} \\ \nabla G_p^\ell(h_{p,t}; \zeta_{p,t}) &= \begin{bmatrix} \nabla F_p^\ell(w_{p,t}; \zeta_{p,t}) V_{p,t}^\ell \\ \nabla F_p^\ell(w_{p,t}; \zeta_{p,t})^\top U_{p,t}^\ell \end{bmatrix}. \end{aligned} \quad (\text{A.4})$$

For simplicity of notation, we denote the stochastic gradients at each iteration by  $G_{p,t}$ , that is

$$G_{p,t} = \begin{cases} \nabla G(h_{p,t}, \zeta_{p,t}) & \text{if } t \bmod E \neq 0 \\ \nabla G(y_{p,t}, \zeta_{p,t}) & \text{if } t \bmod E = 0. \end{cases} \quad (\text{A.5})$$

We note that both  $\nabla G(h_{p,t}; \zeta_{p,t})$  and  $G_{p,t}$  are vectors, even though they are denoted in capital letters.

The size of  $U_{p,t}^\ell$  and  $V_{p,t}^\ell$  are constant at all iterations  $t$ , for any participant  $p$  and layer  $\ell \in \{\rho + 1, \dots, L\}$ . Then, we define  $r(p, \ell)$  to be the rank of  $U_{p,t}^\ell$  and  $V_{p,t}^\ell$  for all  $\ell \in \{\rho + 1, \dots, L\}$ . Given this, for each layer  $\ell \in \{\rho + 1, \dots, L\}$ , we define the maximum layer rank  $r_{\max, \ell} := \max_p(r(p, \ell))$ .

The standard approach for analyzing FL or parallel restarted SGD involves studying the behavior of the average of the iterates, including at iterations where no averaging occurs. However, direct averaging the iterates is challenging because participants may be working with models of varying sizes. Nevertheless, by virtue of Eq. (A.4), it is possible to append matrices of zeros to the factors  $U_p^\ell$  and  $V_p^\ell$  while maintaining the same stochastic gradient iterates. In particular, for a layer  $\ell$  with  $W^\ell \in \mathbb{R}^{m \times n}$ , let

$$\tilde{U}_p^\ell = \begin{bmatrix} U_p^\ell & 0_{m \times (r_{\max, \ell} - r(p, \ell))} \end{bmatrix}, \quad \tilde{V}_p^\ell = \begin{bmatrix} V_p^\ell & 0_{n \times (r_{\max, \ell} - r(p, \ell))} \end{bmatrix}, \quad (\text{A.6})$$

where  $0_{i,j}$  is the zero matrix of size  $i \times j$ .

By padding the factors with properly sized matrices of zeros, the behavior of the iterates is unchanged. In addition, by Eq. (A.6), it follows that

$$\tilde{U}_p^\ell \in \mathbb{R}^{m, r_{\max, \ell}}, \quad \tilde{V}_p^\ell \in \mathbb{R}^{n, r_{\max, \ell}}, \quad \forall p, \forall \ell \in \{\rho + 1, \dots, L\},$$

so that the matrices  $\tilde{U}_p^\ell$  and  $\tilde{V}_p^\ell$  can be directly averaged over the participants  $p$ . In addition, it should be clear that  $W_p^\ell = U_p^\ell (V_p^\ell)^\top = \tilde{U}_p^\ell (\tilde{V}_p^\ell)^\top$ , so that the inclusion of these zeros has no impact on  $W_p^\ell$  as well. Throughout the rest of this section, we omit the tildes for ease of presentation and assume that the factorized representation of participant  $p$ ,  $h_p$ , has been properly padded with zeros to enable direct comparison.

We now define a number of sequences that will be used throughout our analysis. Let the average of the local solutions at time  $t$  be defined as

$$\bar{h}_t := \frac{1}{P} \sum_{p=1}^P h_{p,t}. \quad (\text{A.7})$$

Recalling the definition of  $G_{p,t}$  in Eq. (A.5), it follows that, for any iteration where  $t \bmod E \neq 0$ ,

$$\bar{h}_{t+1} = \bar{h}_t - \frac{\eta}{P} \sum_{p=1}^P G_{p,t}. \quad (\text{A.8})$$

In addition, when  $t \bmod E = 0$ , we define

$$\bar{y}_t := \frac{1}{P} \sum_{p=1}^P y_{p,t}, \quad (\text{A.9})$$

so that

$$\bar{h}_{t+1} = \bar{y}_t - \frac{\eta}{P} \sum_{p=1}^P G_{p,t}. \quad (\text{A.10})$$

Finally, we define the sequence  $\hat{h}_t$  as

$$\hat{h}_t := \begin{cases} \bar{h}_t & \text{if } t \bmod E \neq 0, \\ \bar{y}_t & \text{if } t \bmod E = 0. \end{cases} \quad (\text{A.11})$$

Now, we consider the properties of  $\bar{y}_t$  and  $y_{p,t}$  at any averaging step. For any  $\ell \in \{\rho + 1, \dots, L\}$ , let  $U_t^\ell \Sigma_t^\ell (V_t^\ell)^\top$  be the singular value decomposition of  $W_t^\ell$ , where  $U_t^\ell$  and  $V_t^\ell$  are orthogonal matrices and  $\Sigma_t^\ell$  is the diagonal matrix of the singular values of  $W_t^\ell$ . Then, defining  $\Sigma_{r(p,\ell),t}^\ell = \text{diag}(\sigma_1, \dots, \sigma_{r(p,\ell)}, 0, \dots, 0)$ , for any  $y_{p,t}^\ell$ ,

$$y_{p,t}^\ell = \text{vec} \left( \begin{bmatrix} U_t^\ell \sqrt{\Sigma_{r(p,\ell),t}^\ell} \\ V_t^\ell \sqrt{\Sigma_{r(p,\ell),t}^\ell} \end{bmatrix} \right) \text{ and } \bar{y}_t^\ell = \text{vec} \left( \begin{bmatrix} \frac{1}{P} \sum_{p=1}^P U_t^\ell \sqrt{\Sigma_{r(p,\ell),t}^\ell} \\ \frac{1}{P} \sum_{p=1}^P V_t^\ell \sqrt{\Sigma_{r(p,\ell),t}^\ell} \end{bmatrix} \right).$$

For completeness, we define

$$\bar{h}_0 := \text{vec} \left( \begin{aligned} & \left\{ W^0, \dots, W^\rho, U_0^{\rho+1} \sqrt{\Sigma_0^{\rho+1}}, V_0^{\rho+1} \sqrt{\Sigma_0^{\rho+1}}, \right. \\ & \left. \dots, U_0^L \sqrt{\Sigma_0^L}, V_0^L \sqrt{\Sigma_0^L} \right\} \end{aligned} \right),$$

so that  $g(\bar{h}_0) = f(w_0)$ .

Finally, we use  $\mathbb{E}_{p,t}[\cdot]$  to denote an expected value of the random variable  $\zeta_{p,t}$  conditioned on the full history of randomness up to the start of iteration  $t$ , which we denote by  $\zeta_{[t-1]} := [\zeta_{p,i}]_{p \in \{1, \dots, P\}, i \in \{1, \dots, t-1\}}$ . Specifically, we let

$$\mathbb{E}_{p,t}[\cdot] = \mathbb{E}_{\zeta_{p,t} \sim \mathcal{D}_p}[\cdot | \zeta_{[t-1]}].$$

## A.2. Convergence proofs

**Assumption 4.** (The same as Assumption 1 in the main text) Each function  $f_p(w)$  is continuously differentiable, the gradients of  $f_p(w)$  are Lipschitz with modulus  $L_{\nabla f}$ , and the function  $f(w)$  is bounded below by  $f_{\text{low}} \in \mathbb{R}$ . For each participant  $p$  and iteration  $t$ , the stochastic gradients satisfy

$$\mathbb{E}_{p,t}[\nabla F_p(w_{p,t}; \zeta_{p,t})] = \nabla f_p(w_{p,t}).$$

In addition, there exist constants  $M_f > 0$  and  $\kappa_f > 0$  such that

$$\begin{aligned} \mathbb{E}_{\zeta_p \sim \mathcal{D}_p}[\|\nabla F_p(w; \zeta_p) - \nabla f_p(w)\|^2] &\leq M_f, \quad \forall w, \forall p \\ \mathbb{E}_{\zeta_p \sim \mathcal{D}_p}[\|\nabla F_p(w; \zeta_p)\|^2] &\leq \kappa_f, \quad \forall w, \forall p. \end{aligned}$$

We note here that while the assumption on bounded second moments is strong, it is a common assumption in the literature for non-convex parallel restarted SGD [53].

Recalling Eq. (A.4), it follows that under Assumption 4,

$$\mathbb{E}_{p,t}[\nabla G_p(h_{p,t}; \zeta_{p,t})] = \nabla g_p(h_{p,t}).$$

However, Assumption 4 is insufficient to imply that the gradients of  $g_p$  are Lipschitz continuous gradients nor that the stochastic gradients have bounded variance and bounded second moments. This motivates the following assumption.

**Assumption 5.** (The same as Assumption 2 in the main text) There exists a constant  $\kappa_{UV} > 0$  such that, at every iteration  $t$ ,

$$\|U_{p,t}^\ell\|_F \leq \kappa_{UV}, \quad \|V_{p,t}^\ell\|_F \leq \kappa_{UV}, \quad \forall p, \forall \ell \in \{\rho + 1, \dots, L\},$$

where  $\|\cdot\|_F$  denotes the Frobenius norm.

Under Assumptions 4 and 5, it follows that the gradients of  $g_p(h)$  are Lipschitz continuous for some modulus  $L_{\nabla g} > 0$  and there exist constants  $M_g > 0$  and  $\kappa_g > 0$  such that,

$$\begin{aligned} \mathbb{E}_{\zeta_p \sim \mathcal{D}_p}[\|\nabla G_p(h; \zeta_p) - \nabla g_p(h)\|^2] &\leq M_g, \quad \forall h, \forall p, \\ \mathbb{E}_{\zeta_p \sim \mathcal{D}_p}[\|\nabla G_p(h; \zeta_p)\|^2] &\leq \kappa_g, \quad \forall h, \forall p. \end{aligned}$$

Most of our results will be proven under the previous two assumptions. However, for the final result, we require an additional assumption that allows us to bound the difference between the server model and the low-rank models at each aggregation step.

**Assumption 6.** (The same as Assumption 3 in the main text) There exist constants  $\kappa_{\sigma_1}$  and  $\kappa_{\sigma_2}$  such that, at every iteration  $t$  such that  $t \bmod E = 0$ , we have



$$\mathbb{E} \left[ \left\| \left( \frac{1}{P} \sum_{p=1}^P \sqrt{\Sigma_{r(p,\ell),t}^\ell} \right) \left( \frac{1}{P} \sum_{p=1}^P \sqrt{\Sigma_{r(p,\ell),t}^\ell} \right) - \Sigma_t^\ell \right\|_F^2 \right] \leq \kappa_{\sigma_1}$$

and  $\mathbb{E} \left[ \left\| \frac{1}{P} \sum_{i=1}^P \sqrt{\Sigma_{r(i,\ell),t}^\ell} - \sqrt{\Sigma_{r(p,\ell),t}^\ell} \right\|_F^2 \right] \leq \kappa_{\sigma_2}$

In addition, the function  $f$  is Lipschitz constant with modulus  $L_f$ .

Recalling Eq. (A.1), the first inequality in this assumption assumes that the expected norm difference between  $\text{inv}(\bar{y}_t)^\ell$  and  $w_t^\ell$  is bounded, while the second assumes that, in expectation, the norm difference of the factors of  $y_{p,t}^\ell$  and  $\bar{y}_t^\ell$  are also bounded. Essentially, these two inequalities state that the expected difference between the “true” model and the average of the low-rank models is bounded, as well as the difference between the average of the low-rank models and any participant’s low-rank model. We remark without such an assumption, there is no reason to expect that Algorithm 1 would produce a reasonable model, as the arbitrary error may be added during the network factorization step.

We begin with several preliminary lemmas and note that the structure of our analysis is heavily based on the analysis of parallel restarted SGD found in [53]. First, we derive a preliminary result about the drift between individual models  $h_{p,t}$  and the model average  $\bar{h}_t$ .

**Lemma 1.** Let Assumptions 4 and 5 hold. Then, for any participant  $p$  and  $t > 0$ ,

$$\mathbb{E}[\|\bar{h}_t - h_{p,t}\|^2] \leq 6\eta^2 E^2 \kappa_g + 3\mathbb{E}[\|\bar{y}_{t_0} - y_{p,t_0}\|^2],$$

where  $t_0$  is the largest  $t_0 < t$  such that  $t_0 \bmod E = 0$ .

Next, we give a preliminary lemma that bounds the expected difference between the average sequence  $\bar{h}$  at subsequent iterates.

**Lemma 2.** Let Assumptions 4 and 5. Then, when  $t \bmod E \neq 0$ ,

$$\mathbb{E}[\|\bar{h}_{t+1} - \bar{h}_t\|^2] \leq \frac{\eta^2 M_g}{P} + \eta^2 \mathbb{E} \left[ \left\| \frac{1}{P} \sum_{p=1}^P \nabla g_p(h_{p,t}) \right\|^2 \right].$$

Otherwise, when  $t \bmod E = 0$ ,

$$\mathbb{E}[\|\bar{h}_{t+1} - \bar{y}_t\|^2] \leq \frac{\eta^2 M_g}{P} + \eta^2 \mathbb{E} \left[ \left\| \frac{1}{P} \sum_{p=1}^P \nabla g_p(y_{p,t}) \right\|^2 \right].$$

Next, we provide a bound on the expectation of the inner product of the average gradient and the difference of the averaged iterates.

**Lemma 3.** Let Assumptions 4 and 5. Then, for any iteration  $t$  such that  $t \bmod E \neq 0$ ,

$$\begin{aligned} \mathbb{E}[\nabla g(\bar{h}_t)^\top (\bar{h}_{t+1} - \bar{h}_t)] &\leq -\frac{\eta}{2} \mathbb{E}[\|\nabla g(\bar{h}_t)\|^2] - \frac{\eta}{2} \mathbb{E} \left[ \left\| \frac{1}{P} \sum_{p=1}^P \nabla g_p(h_{p,t}) \right\|^2 \right] \\ &\quad + 3\eta^3 L_{\nabla g}^2 \kappa_g E^2 + \frac{3\eta L_{\nabla g}^2}{2P} \sum_{p=1}^P \mathbb{E}[\|\bar{y}_{t_0} - y_{p,t_0}\|^2], \end{aligned}$$

where  $t_0$  is the largest  $t_0 < t$  such that  $t_0 \bmod E = 0$ . In addition, when  $t \bmod E = 0$ ,

$$\begin{aligned} E[\nabla g(\bar{y}_t)^\top (\bar{h}_{t+1} - \bar{y}_t)] &\leq -\frac{\eta}{2} \mathbb{E}[\|\nabla g(\bar{y}_t)\|^2] - \frac{\eta}{2} \mathbb{E} \left[ \left\| \frac{1}{P} \sum_{p=1}^P \nabla g_p(y_{p,t}) \right\|^2 \right] \\ &\quad + \frac{\eta L_{\nabla g}^2}{2P} \sum_{p=1}^P \mathbb{E}[\|\bar{y}_t - y_{p,t}\|^2]. \end{aligned}$$

Next, we present a general convergence result under Assumptions 4 and 5. In the statement of the theorem, we use the set  $\mathcal{E}$  to denote the set of iterations at which Algorithm 1 averages (except for the final output averaging), i.e.

$$\mathcal{E} := \{0, E, \dots, (E/T) - E\}.$$

**Table B.8**

The hybrid ResNet-18 architecture for CIFAR-10 dataset in the experiments.

Layer Name	ResNet-18	Low Rank ResNet-18
conv1	3×3, 64, stride 1, padding 1	3×3, 64, stride 1, padding 1
layer1	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix}, \begin{bmatrix} conv_u(64, 64 * r, 3, 1) \\ conv_v(64 * r, 64, 1, 3) \end{bmatrix} \times 2$
layer2	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} conv_u(128, 128 * r, 3, 1) \\ conv_v(128 * r, 128, 1, 3) \end{bmatrix} \times 4$
layer3	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} conv_u(256, 256 * r, 3, 1) \\ conv_v(256 * r, 256, 1, 3) \end{bmatrix} \times 4$
layer4	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} conv_u(512, 512 * r, 3, 1) \\ conv_v(512 * r, 512, 1, 3) \end{bmatrix} \times 4$

**Table B.9**

The hybrid ResNet-34 architecture for CIFAR-100 and Tiny-ImageNet datasets in the experiments.

Layer Name	ResNet-34	Low Rank ResNet-34
conv1	3×3, 64, stride 1, padding 1	3×3, 64, stride 1, padding 1
layer1	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 3$
layer2	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 4$
layer3	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} conv_u(256, 256 * r, 3, 1) \\ conv_v(256 * r, 256, 1, 3) \end{bmatrix} \times 12$
layer4	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} conv_u(512, 512 * r, 3, 1) \\ conv_v(512 * r, 512, 1, 3) \end{bmatrix} \times 6$

We recall that by the definition of Algorithm 1,  $T \bmod E = 0$ , so that  $E/T$  is an integer value, so this set is well defined.

**Theorem 2.** Let Assumptions 4 and 5 hold. Let  $0 < \eta \leq \frac{1}{L_{\nabla g}}$ . Then,

$$\begin{aligned} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}[\|\nabla g(\hat{h}_t)\|^2] &\leq \frac{2}{T\eta} (f(w_0) - f_{low}) + 6\eta^2 L_{\nabla g}^2 \kappa_g E^2 + \frac{\eta L_{\nabla g} M_g}{P} \\ &+ \frac{2}{T\eta} \sum_{t \in \mathcal{E}} \mathbb{E}[g(\bar{y}_t) - g(\bar{h}_t)] + \frac{3L_{\nabla g}^2 E}{TP} \sum_{t \in \mathcal{E}} \sum_{p=1}^P \mathbb{E}[\|\bar{y}_t - y_{p,t}\|^2]. \end{aligned}$$

Now, we present our main result, which follows under the addition of Assumption 6.

**Theorem 3.** Let Assumptions 4, 5, and 6 hold and let  $\hat{h}_t$  be defined in Eq. (A.11). Then,

$$\begin{aligned} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}[\|\nabla g(\hat{h}_t)\|^2] &\leq \frac{2}{T\eta} (f(w_0) - f_{low}) + \frac{2L_f(L - \rho)\kappa_{\sigma_1}}{E\eta} \\ &+ 6\eta\kappa_g E \left( \eta L_{\nabla g}^2 E + 2L_f \kappa_{UV}^2 \right) + \frac{\eta L_{\nabla g} M_g}{P} + (6(L - \rho)\kappa_{\sigma_2}) \left( L_{\nabla g}^2 + \frac{2L_f \kappa_{UV}^2}{E\eta} \right). \end{aligned}$$

## Appendix B. Hybrid network structure

Tables B.8 and B.9 summarize the model architecture used in our experiments. Fig. B.9 presents the number of parameters and MACs of both ResNet-18 and ResNet-34 across various datasets. We find that the later layers (Layer3 and Layer4) contribute the most parameters, i.e., over 93% for ResNet-34 and ResNet-18, and the two later layers also contribute 55.6% computation overhead on ResNet-34. From Fig. B.9, we can conclude that compressing the later layers can significantly reduce the communication cost and computation overhead.

## Data availability

No data was used for the research described in the article.

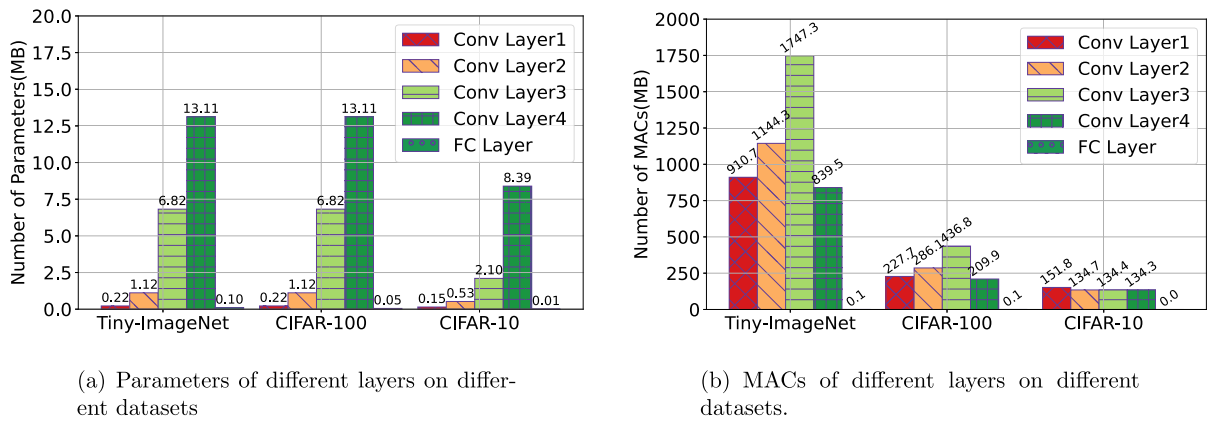


Fig. 9. Parameters and MACs of different layers of ResNet on various datasets.

## References

- [1] B. McMahan, E. Moore, D. Ramage, S. Hampson, B.A. y Arcas, Communication-efficient learning of deep networks from decentralized data, in: Proc. of AISTATS, 2017, pp. 1273–1282.
- [2] W.Y.B. Lim, J.S. Ng, Z. Xiong, J. Jin, Y. Zhang, D. Niyato, C. Leung, C. Miao, Decentralized edge intelligence: a dynamic resource allocation framework for hierarchical federated learning, IEEE Trans. Parallel Distrib. Syst. 33 (3) (2021) 536–550.
- [3] L.L. Pilla, Optimal task assignment for heterogeneous federated learning devices, in: Proc. of IPDPS, 2021, pp. 661–670.
- [4] G. Wang, B. Gu, Q. Zhang, X. Li, B. Wang, C. Ling, A unified solution for privacy and communication efficiency in vertical federated learning, in: Proc. of the 37th Conference on Neural Information Processing Systems, 2023, pp. 13480–13491.
- [5] P. Kairouz, H.B. McMahan, B. Avent, A. Bellet, M. Bennis, A.N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, et al., Advances and open problems in federated learning, Found. Trends Mach. Learn. 14 (1–2) (2021) 1–210.
- [6] J. Wang, Z. Charles, Z. Xu, G. Joshi, H.B. McMahan, M. Al-Shedivat, G. Andrew, S. Avestimehr, K. Daly, D. Data, et al., A field guide to federated optimization, arXiv preprint, arXiv:2107.06917, 2021.
- [7] Z. Gao, X. Chen, X. Shao, Robust federated learning for edge-intelligent networks, Sci. China Inf. Sci. 65 (3) (2022) 1–9.
- [8] F. Lai, X. Zhu, H.V. Madhyastha, M. Chowdhury, Oort: efficient federated learning via guided participant selection, in: Proc. of OSDI, 2021, pp. 19–35.
- [9] J. Sun, T. Chen, G.B. Giannakis, Q. Yang, Z. Yang, Lazily aggregated quantized gradient innovation for communication-efficient federated learning, IEEE Trans. Pattern Anal. Mach. Intell. 44 (4) (2022) 2031–2044.
- [10] J. Wolfraht, N. Sreekumar, D. Kumar, Y. Wang, A. Chandra, Haccs: heterogeneity-aware clustered client selection for accelerated federated learning, in: Proc. of IPDPS, 2022, pp. 985–995.
- [11] J. Wang, X. Yang, S. Cui, L. Che, L. Lyu, D. Xu, F. Ma, Towards personalized federated learning via heterogeneous model reassembly, in: Proc. of NeurIPS, 2023.
- [12] T. Nishio, R. Yonetani, Client selection for federated learning with heterogeneous resources in mobile edge, in: Proc. of International Conference on Communications (ICC), 2019, pp. 1–7.
- [13] A. Sultana, M.M. Haque, L. Chen, F. Xu, X. Yuan, Eiffel: efficient and fair scheduling in adaptive federated learning, IEEE Trans. Parallel Distrib. Syst. 33 (12) (2022) 4282–4294.
- [14] D. Li, J. Wang, FedMD: heterogeneous federated learning via model distillation, arXiv preprint, arXiv:1910.03581, 2019.
- [15] T. Lin, L. Kong, S.U. Stich, M. Jaggi, Ensemble distillation for robust model fusion in federated learning, in: Proc. of NeurIPS, 2020, pp. 2351–2363.
- [16] S. Itahara, T. Nishio, Y. Koda, M. Morikura, K. Yamamoto, Distillation-based semi-supervised federated learning for communication-efficient collaborative training with non-iid private data, IEEE Trans. Mob. Comput. 22 (1) (2021) 191–205.
- [17] Y.J. Cho, A. Manoel, G. Joshi, R. Sim, D. Dimitriadis, Heterogeneous ensemble knowledge transfer for training large models in federated learning, in: Proc. of IJCAI, 2022, pp. 2881–2887.
- [18] G. Hinton, O. Vinyals, J. Dean, Distilling the knowledge in a neural network, arXiv preprint, arXiv:1503.02531, 2015.
- [19] S. Stanton, P. Izmailov, P. Kirichenko, A.A. Alemi, A.G. Wilson, Does knowledge distillation really work?, Adv. Neural Inf. Process. Syst. 34 (2021) 6906–6919.
- [20] E. Diao, J. Ding, V. Tarokh, HeteroFL: computation and communication efficient federated learning for heterogeneous clients, in: Proc. of ICLR, 2020.
- [21] J. Hong, H. Wang, Z. Wang, J. Zhou, Efficient split-mix federated learning for on-demand and in-situ customization, in: Proc. of ICLR, 2022.
- [22] S. Alam, L. Liu, M. Yan, M. Zhang, Fedrolex: model-heterogeneous federated learning with rolling sub-model extraction, Adv. Neural Inf. Process. Syst. 35 (2022) 29677–29690.
- [23] E.L. Denton, W. Zaremba, J. Bruna, Y. LeCun, R. Fergus, Exploiting linear structure within convolutional networks for efficient evaluation, Adv. Neural Inf. Process. Syst. 27 (2014).
- [24] E.J. Hu, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, W. Chen, et al., LoRA: low-rank adaptation of large language models, in: Proc. of ICLR, 2021.
- [25] Y.-L. Sung, J. Cho, M. Bansal, VL-ADAPTER: parameter-efficient transfer learning for vision-and-language tasks, in: Proc. of CVPR, 2022, pp. 5227–5237.
- [26] Y. Mao, L. Mathias, R. Hou, A. Almahairi, H. Ma, J. Han, W.-t. Yih, M. Khabsa, UniPELT: a unified framework for parameter-efficient language model tuning, in: Proc. of ACL, 2022, pp. 6253–6264.
- [27] L. Yi, G. Wang, X. Liu, Z. Shi, H. Yu, Fedgh: Heterogeneous federated learning with generalized global header, in: Proc. of the 31st ACM International Conference on Multimedia, 2023, pp. 8686–8696.
- [28] J. Zhang, C. Chen, B. Li, L. Lyu, S. Wu, S. Ding, C. Shen, C. Wu, Dense: Data-free one-shot federated learning, Adv. Neural Inf. Process. Syst. 35 (2022) 21414–21428.
- [29] S. Wang, Y. Fu, X. Li, Y. Lan, M. Gao, et al., DFRD: data-free robustness distillation for heterogeneous federated learning, in: Proc. of the 37th International Conference on Neural Information Processing Systems, 2024, pp. 17854–17866.
- [30] S. Horvath, S. Laskaridis, M. Almeida, I. Leontiadis, S. Venieris, N. Lane, FJORD: fair and accurate federated learning under heterogeneous targets with ordered dropout, in: Proceedings of the 35th International Conference on Neural Information Processing Systems, 2021, pp. 12876–12889.
- [31] J. Yu, T.S. Huang, Universally slimmable networks and improved training techniques, in: Proc. of CVPR, 2019, pp. 1803–1811.
- [32] V. Lebedev, Y. Ganin, M. Rakhuba, I.V. Oseledets, V.S. Lempitsky, Speeding-up convolutional neural networks using fine-tuned cp-decomposition, in: Proc. of ICLR (Poster), 2015.

- [33] C. Tai, T. Xiao, Y. Zhang, X. Wang, E. Weinan, Convolutional neural networks with low-rank regularization, in: 4th International Conference on Learning Representations, ICLR 2016, 2016.
- [34] A.-H. Phan, K. Sobolev, K. Sozykin, D. Ermilov, J. Gusak, P. Tichavský, V. Glukhov, I. Oseledets, A. Cichocki, Stable low-rank tensor decomposition for compression of convolutional neural network, in: Proc. of ECCV, 2020, pp. 522–539.
- [35] Y. Wang, C. Xu, C. Xu, D. Tao, Beyond filters: compact feature map for portable deep model, in: International Conference on Machine Learning, PMLR, 2017, pp. 3703–3711.
- [36] T.N. Sainath, B. Kingsbury, V. Sindhwani, E. Arisoy, B. Ramabhadran, Low-rank matrix factorization for deep neural network training with high-dimensional output targets, in: Proc. of ICASSP, 2013, pp. 6655–6659.
- [37] J. Xue, J. Li, Y. Gong, Restructuring of deep neural network acoustic models with singular value decomposition, in: Proc. of INTERSPEECH, 2013, pp. 2365–2369.
- [38] J. Konečný, H.B. McMahan, F.X. Yu, P. Richtárik, A.T. Suresh, D. Bacon, Federated learning: strategies for improving communication efficiency, arXiv preprint, arXiv:1610.05492, 2016.
- [39] Z. Qiao, X. Yu, J. Zhang, K.B. Letaief, Communication-efficient federated learning with dual-side low-rank compression, arXiv preprint, arXiv:2104.12416, 2021.
- [40] N. Hyeon-Woo, M. Ye-Bin, T.-H. Oh, FedPara: low-rank Hadamard product for communication-efficient federated learning, in: Proc. of ICLR, 2022.
- [41] W. Jeong, S.J. Hwang, Factorized-fl: personalized federated learning with parameter factorization & similarity matching, Adv. Neural Inf. Process. Syst. 35 (2022) 35684–35695.
- [42] M.G. Arivazhagan, V. Aggarwal, A.K. Singh, S. Choudhary, Federated learning with personalization layers, arXiv preprint, arXiv:1912.00818, 2019.
- [43] A. Fallah, A. Mokhtari, A. Ozdaglar, Personalized federated learning: a meta-learning approach, arXiv preprint, arXiv:2002.07948, 2020.
- [44] C.T. Dinh, N. Tran, J. Nguyen, Personalized federated learning with Moreau envelopes, Adv. Neural Inf. Process. Syst. 33 (2020) 21394–21405.
- [45] J. Chen, W. Xu, S. Guo, J. Wang, J. Zhang, H. Wang, Fedtune: a deep dive into efficient federated fine-tuning with pre-trained transformers, arXiv preprint, arXiv:2211.08025, 2022.
- [46] Z. Zhang, Y. Yang, Y. Dai, Q. Wang, Y. Yu, L. Qu, Z. Xu, Fedpetuning: when federated learning meets the parameter-efficient tuning methods of pre-trained language models, in: Annual Meeting of the Association for Computational Linguistics 2023, Association for Computational Linguistics (ACL), 2023, pp. 9963–9977.
- [47] N. Houshy, A. Giurigu, S. Jastrzebski, B. Morroni, Q. De Laroussilhe, A. Gsmundo, M. Attariyan, S. Gelly, Parameter-efficient transfer learning for nlp, in: International Conference on Machine Learning, PMLR, 2019, pp. 2790–2799.
- [48] X.L. Li, P. Liang, Prefix-tuning: optimizing continuous prompts for generation, in: Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), 2021, pp. 4582–4597.
- [49] M. Khodak, N.A. Tenenholz, L. Mackey, N. Fusi, Initialization and regularization of factorized neural layers, in: Proc. of ICLR, 2020.
- [50] H. Wang, S. Agarwal, D. Papailiopoulos, Pufferfish: communication-efficient models at no extra cost, in: Proc. of MLSys, 2021, pp. 365–386.
- [51] V. Mothukuri, R.M. Parizi, S. Pouriyeh, Y. Huang, A. Dehghantanha, G. Srivastava, A survey on security and privacy of federated learning, Future Gener. Comput. Syst. 115 (2021) 619–640.
- [52] M. Rigaki, S. Garcia, A survey of privacy attacks in machine learning, ACM Comput. Surv. 56 (4) (2023) 1–34.
- [53] H. Yu, S. Yang, S. Zhu, Parallel restarted sgd with faster convergence and less communication: demystifying why model averaging works for deep learning, in: Proc. of AAAI, 2019, pp. 5693–5700.
- [54] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, in: Proc. of NeurIPS, 2017, pp. 5998–6008.
- [55] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al., An image is worth 16x16 words: transformers for image recognition at scale, in: Proc. of ICLR, 2020.
- [56] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, B. Guo, Swin transformer: hierarchical vision transformer using shifted windows, in: Proc. of ICCV, 2021, pp. 10012–10022.
- [57] L. Qu, Y. Zhou, P.P. Liang, Y. Xia, F. Wang, E. Adeli, L. Fei-Fei, D. Rubin, Rethinking architecture design for tackling data heterogeneity in federated learning, in: Proc. of CVPR, 2022, pp. 10061–10071.
- [58] A. Krizhevsky, Learning multiple layers of features from tiny images, Master's thesis, Department of Computer Science, University of Toronto, 2009.
- [59] T.-M.H. Hsu, H. Qi, M. Brown, Measuring the effects of non-identical data distribution for federated visual classification, arXiv preprint, arXiv:1909.06335, 2019.
- [60] B. Lester, R. Al-Rfou, N. Constant, The power of scale for parameter-efficient prompt tuning, in: Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, 2021, pp. 3045–3059.
- [61] S. Babakniya, A.R. Elkordy, Y.H. Ezzeldin, Q. Liu, K.-B. Song, M. El-Khamy, S. Avestimehr, Slora: federated parameter efficient fine-tuning of language models, arXiv preprint, arXiv:2308.06522, 2023.