Heterogeneous LoRA for Federated Fine-tuning of On-Device Foundation Models

Yae Jee Cho[1,2]*, **Luyang Liu**[2], **Zheng Xu**[2], **Aldi Fahrezi**[2], **Gauri Joshi**[1]
[1]Carnegie Mellon University, [2]Google Research
{yaejeecho,luyangliu,xuzheng,aldifahrezi}@google.com,
gaurij@andrew.cmu.edu

Abstract

Foundation models (FMs) adapt surprisingly well to downstream tasks with fine-tuning. However, their colossal parameter space prohibits their training on resource-constrained edge-devices. For federated fine-tuning, we need to consider the smaller FMs of few billion parameters at most, namely *on-device FMs (ODFMs)*, which can be deployed on-device. Federated fine-tuning of ODFMs has unique challenges non-present in standard fine-tuning: i) ODFMs poorly generalize to downstream tasks due to their limited sizes making proper fine-tuning imperative to their performance, and ii) devices have limited and heterogeneous system capabilities and data that can deter the performance of fine-tuning.

Tackling these challenges, we propose HETLORA, a feasible and effective federated fine-tuning method for ODFMs that leverages the system and data heterogeneity at the edge. HETLORA allows *heterogeneous LoRA ranks* across clients for their individual system resources, and efficiently aggregates and distributes these LoRA modules in a data-aware manner by applying rank self-pruning locally and sparsity-weighted aggregation at the server. It combines the advantages of high and low-rank LoRAs, achieving improved convergence speed and final performance compared to homogeneous LoRA. Furthermore, HETLORA has enhanced computation and communication efficiency compared to full fine-tuning making it more feasible for the edge.

1 Introduction

The emerging foundation models (FMs), easily ranging from few to hundreds of billions of parameters (Gemini Team Google, 2024; Bommasani et al., 2022; Radford et al., 2021; Devlin et al., 2019; OpenAI, 2023; Google, 2022; Touvron et al., 2023; Brown et al., 2020; Google, 2022; Driess et al., 2023), have remarkable zero/few shot learning capabilities. They often perform surprisingly well on a variety of tasks such as text/image generation, language translation, and conversing in natural language with fine-tuning. Such standard FMs, however, demand costly resources for directly fine-tuning their entire parameter space. To tackle this issue, many works have proposed different parameter-efficient fine-tuning (PEFT) methods such as prompt tuning (Lester et al., 2021), utilizing adapters (Houlsby et al., 2019), or low-rank adaptation (LoRA) of the original model (Hu et al., 2021) which freezes the original pre-trained parameters of the FM and train additional, smaller number of parameters instead.

Unfortunately, such PEFT methods rely on the assumptions that i) FMs are deployed to and trained with the data of a *single* client for adaptation to the downstream task and ii) the client has enough resources to even fit a standard FM of hundred billion size for, at least, inference. In practice, more often than not we are interested in fine-tuning FMs for on-device private data that is distributed across multiple devices, and these clients at the edge rarely have such resources to even fit standard FMs for inference. One of the examples is to use FMs for downstream tasks that requires training with private, non-sharable data such as medical or law-related documents distributed across clients (Manoel et al., 2023; Shoham and Rappoport, 2023; Zhang et al., 2023c). In such cases, fine-tuning of the FMs needs to be brought to the edge that has limited system resources.

Federated Fine-Tuning & ODFMs. We consider such *federated fine-tuning* scenarios, where

*Work done while interning at Google Research. Corresponding authors: {yaejeecho,luyangliu}@google.com

Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, pages 12903–12913
November 12-16, 2024 ©2024 Association for Computational Linguistics

	Zero-Shot	Few-Shot	Full-Training
PaLM 2 XXS	2930.23	2541.86	**23.71**
PaLM 2 XS	2712.86	481.95	**18.32**

Table 1: Perplexity of PaLM 2 for zero-shot, few-shot (5 communication rounds), and full federated fine-tuning (200 communication rounds) for chat response on the multi-session chat data (further experimental details are in Section 4.)

we train a set of parameters collaboratively across clients to obtain a global set of parameters that can be plugged in to the FM for the targeted downstream task. Note that federated fine-tuning is orthogonal to personalization of FMs in federated learning (FL) that has been explored in other previous work (Guo et al., 2023; Wu et al., 2024), which aims to train parameters that perform well for individual clients rather than general downstream tasks. We also define *on-device FMs (ODFMs)* as models with few billion parameters at max that are able to fit into memory on limited capacity clients considering current hardwares.

Challenges. Federated fine-tuning of ODFMs entails two major challenges non-present in neither standard PEFT of FMs nor federated training of non-FMs. First, *ODFMs poorly generalize to downstream tasks.* Unlike ODFMs, standard FMs have their zero/few-shot learning capability by their large parameter space trained on massive data. However, as we show in Table 1 and as previous literature has shown (Kojima et al., 2022; Lester et al., 2021), FMs' performance deteriorates as their sizes get smaller and federated fine-tuning is not merely useful but *inevitable* for ODFMs to perform well for on-device downstream tasks.

Second, *devices have limited and heterogeneous system capabilities and data distributions* (Wang et al., 2019; Bonawitz et al., 2016; Sahu et al., 2020). Without suitable PEFT methods that flexibly adapts to such heterogeneity across devices, we would only have limited performance output from federated fine-tuning of ODFMs. For instance, with homogeneous rank deployment in LoRA that is agnostic of system heterogeneity, the client with the least system resource becomes the bottleneck, forcing a smaller rank to be deployed across *all* clients despite the other clients being able to handle higher ranks (see Table 2 for the effect of rank on the trainable parameter #). Additionally, data heterogeneity causing model drifts can make the model converge to a suboptimal point.

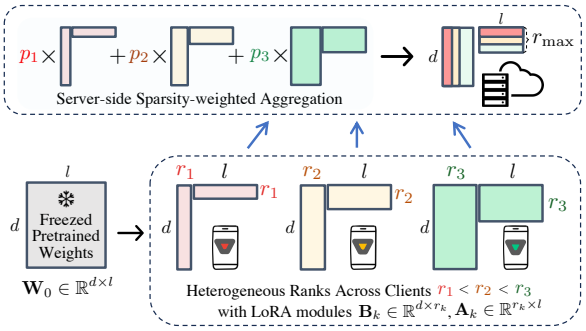


Figure 1: Overview of heterogeneous rank deployment of LoRA: the pretrained weights $\mathbf{W}_0$ are stored on-device and heterogeneous ranks are assigned to different clients with $r_{\text{min}} = r_1 < r_2 < r_3 = r_{\text{max}}$. In our proposed HETLORA, the server receives the trained heterogeneous LoRA modules and aggregates them with sparsity-weighted aggregation to update the global LoRA module.

Contributions. In our work, we push the limits of federated fine-tuning of ODFMs by proposing HETLORA that is aware of heterogeneity residing at the edge. Our main flagship results (in Table 3 and Fig. 5), show that our proposed HETLORA outperforms the standard LoRA method and achieves comparable performance with full fine-tuning with a significantly smaller number of trained parameters. We verify HETLORA with PaLM 2 (Google, 2023) of XXS and XS size for chat responses on the multi-session chat data (Xu et al., 2021) and text summarization for the Reddit data (Völske et al., 2017), both real world data from clients. Our contributions are summarized as:

- We propose HETLORA that can apply different rank LoRA modules to different clients to cater to the heterogeneous system capabilities and data complexities of the clients, via utilizing rank self-pruning and sparsity-weighted aggregation.
- We show the performance of naïvely applying LoRA with homogeneous ranks across clients for federated fine-tuning, and show that while large ranks help in speeding-up training, they lead to faster overfitting while smaller ranks are slower in training but does not suffer from overfitting.
- We then evaluate our proposed HETLORA to show that it outperforms naïvely applying homogeneous ranks across clients in terms of both training speed, communication/computation efficiency, and final performance.

	$r=1$	$r=5$	$r=10$	$r=20$	$r=50$	$r=100$	$r=150$	$r=200$
PaLM 2 XXS, PaLM 2 XS	0.02%	0.11%	0.21%	0.42%	1.05%	2.10%	3.14%	4.19%

Table 2: Percentage of the LoRA parameters' size for different ranks r compared to the original pre-trained ODFM's parameter size. Even for large ranks such as $r = 200$ the trainable LoRA parameters' size compared to the original pre-trained ODFM size is less than 5% for both PaLM 2-XS and PaLM 2-XXS.

2 Related Work

Parameter-Efficient Fine Tuning. Most PEFT methods either train a subset of parameters within the existing FM whilst other parameters are freezed or introduce an additional set of trainable parameters whilst keeping the original FM freezed. For the former, methods such as head or bias fine-tuning (Wei et al., 2021; Bu et al., 2022; Lee et al., 2019; Zaken et al., 2021) has been explored, and for the latter, methods such as adapters (Houlsby et al., 2019), prompt (Lester et al., 2021) or prefix-tuning (Li and Liang, 2021), and low-rank approximation (Hu et al., 2021) has been proposed. While these number of methods has been proven to perform as well as full model fine-tuning with just a few number of parameters for the centralized setting, it has not been thoroughly explored how these methods would perform for a much smaller FM such as ODFMs, in the decentralized setting where clients' system-capacities and data can be heterogeneous and much limited.

Federated Fine-Tuning. Several recent work has proposed to combine the PEFT methods devised for the centralized setting to FL (Zhou et al., 2023; Yu et al., 2023) such as training prompts or adapters collaboratively with FL (Guo et al., 2022; Chen et al., 2022; Zhang et al., 2023a; Shysheya et al., 2023; Legate et al., 2023). Another line of work has proposed to perform a few-shot or nearly zero-shot training of FMs with FL for improved communication-efficiency (Wortsman et al., 2023; Zhang et al., 2023d). However, these work either overlooks that most devices do not have the resource to fit standard FMs (Touvron et al., 2023; Brown et al., 2020) even for inference or does not consider the heterogeneous system capacities of the clients. It is detrimental to consider these factors since FMs that actually fits to the devices in FL are much smaller, making them weaker in the general intelligence capabilities, and also heterogeneous system capacities can prohibit deploying same sized PEFT parameters across clients.

LoRA and FL. There has been a number of variations of LoRA (Zhang et al., 2023b; Liu et al., 2024; Sun et al., 2024; Horváth et al., 2024) from its first proposal (Hu et al., 2021). However, only a few number of recent work has looked in to using LoRA for FL. In Babakniya et al. (2023), the importance of the initialization for the LoRA modules is evaluated where they propose to train the LoRA modules with FL and then perform singular value decomposition (SVD) to gain a good initialization of the LoRA modules. However, the training process of LoRA itself is not altered to adapt to heterogeneous system capabilieis of devices. Another recent work (Yi et al., 2023) has evaluated LoRA in the context of personalized FL, but other than applying LoRA to personalization, the LoRA method itself is, again, not changed. To the best of our knowledge, our work is the first method to consider both data and system heterogeneity for improving training speed, communication/computation efficiency, and final performance of ODFM's federated fine-tuning. Contemporary approaches (Bai et al., 2024; Anonymous, 2024) have proposed to use SVD to decompose the full matrix to deploy heterogeneous ranks across clients. We compare our proposed method with using SVD for heterogeneous LoRA in our work and show that the performance is underwhelming compared to our proposed heterogeneous LoRA.

3 Federated Fine-Tuning with LoRA

3.1 Preliminaries

Formally, we define the pre-trained ODFM as $\mathbf{W}_0 \in \mathbb{R}^{d\times l}$ and the trainable low-rank decomposed matrix as $\Delta\mathbf{W} \in \mathbb{R}^{d\times l}$. In standard LoRA (Hu et al., 2021) under the centralized setting, the low-rank decomposition of $\Delta\mathbf{W}$ is constructed such that $\Delta\mathbf{W} = \mathbf{BA}$ where $\mathbf{B} \in \mathbb{R}^{d\times r}$ and $\mathbf{A} \in \mathbb{R}^{r\times l}$ are the low rank decomposition of $\Delta\mathbf{W}$ with identical rank r. Now, let us consider LoRA for federated fine-tuning where there are M total clients. Each client $k \in [M]$ has pri-

vate data $\mathcal{B}_k$ and its corresponding local empirical loss function $F_k(\mathbf{W}) = \frac{1}{|\mathcal{B}_k|} \sum_{\xi \in \mathcal{B}_k} \ell(\mathbf{W}, \xi)$, where $\ell(\mathbf{W}, \xi)$ is the loss for model $\mathbf{W}$ at data sample ξ. The optimization task for federated fine-tuning is to collaboratively find the global parameters which we define as $\overline{\mathbf{B}}$ and $\overline{\mathbf{A}}$, given the pretrained knowledge $\mathbf{W}_0$ that can minimize the global objective $F(\overline{\mathbf{W}}) = \frac{1}{M} \sum_{k=1}^{M} F_k(\overline{\mathbf{W}})$ where $\overline{\mathbf{W}} = \mathbf{W}_0 + \overline{\mathbf{B}}\,\overline{\mathbf{A}}$. Later in the paper, when introducing heterogeneous LoRA we truncate the LoRA modules' rank dimension, for example from $\mathbf{B} \in \mathbb{R}^{d \times r}$, $\mathbf{A} \in \mathbb{R}^{r \times l}$ to $\mathbf{B}' \in \mathbb{R}^{d \times r'}$, $\mathbf{A}' \in \mathbb{R}^{r' \times l}$ where $r' < r$. Throughout the paper, we denote such truncation of a matrix with the : symbol for each row and column at the subscript. For instance, for truncation to $r' < r$ at the column for the matrix $\mathbf{B} \in \mathbb{R}^{d \times r}$, we keep all the columns until r' and omit the last $r - r'$ columns and denote the resulting matrix it as $\mathbf{B}_{:,:r'}$.

3.2 Naïve Case: Homogeneous LoRA

A straightforward way to perform federated fine-tuning with LoRA is to train the LoRA modules $\mathbf{B}$, $\mathbf{A}$ with homogeneous rank r across all clients with standard FL (McMahan et al., 2017). Specifically, first the clients have the pre-trained ODFM weights $\mathbf{W}_0$ stored in their devices prior to training for the forward pass when training the LoRA modules. Then, the server sends the global LoRA modules $\overline{\mathbf{B}}^{(t)}$, $\overline{\mathbf{A}}^{(t)}$ to the set of m selected clients $\mathcal{S}^{(t)}$ per communication round t. Each selected client $k \in \mathcal{S}^{(t)}$ trains the LoRA modules on their local data for a few local iterations (usually with mini-batch SGD) and send the updated modules $\mathbf{B}_k^{(t)}$, $\mathbf{A}_k^{(t)}$ back to the server. The server then updates the global LoRA modules accordingly to $\overline{\mathbf{B}}^{(t+1)} = \sum_{k \in \mathcal{S}^{(t)}} \mathbf{B}_k^{(t)}/m$, $\overline{\mathbf{A}}^{(t+1)} = \sum_{k \in \mathcal{S}^{(t)}} \mathbf{A}_k^{(t)}/m$ and sends back to the next set of selected clients for the next communication round. This training process is nearly identical to the standard FL algorithm (McMahan et al., 2017) except that the pretrained weights $\mathbf{W}_0$ are freezed and locally stored in the clients' devices and only the LoRA moduels are trained and communicated.

Instead of such homogeneous rank deployment across all clients, it is not only possible but even more practical and feasible to use heterogeneous rank deployment for federated fine-tuning (we later show that this also improves final performance of the ODFMs). This involves training LoRA modules with varying ranks across clients, based on their system capabilities. This allows a more practical setting for federated fine-tuning of ODFMs where, for instance, the client with the smallest system capacity does not have to become the bottleneck that forces deploying a smaller rank across *all clients*. Instead, with heterogeneous rank deployment, each client can use a rank that suits for itself. However, heterogeneous rank deployment poses challenges in aggregating and redistributing the LoRA modules. To address these challenges, we introduce a solution called HETLORA, which pushes the limits beyond homogeneous LoRA.

3.3 Proposed Method: Heterogeneous LoRA

Overview. Our proposed heterogeneous LoRA method, namely HETLORA, is not restricted to any specific method to assign the ranks to the clients and the clients can decide their respective ranks themselves. For formality, in our paper, we formulate that each client has a rank denoted as r_k, within a range of $r_k \in [r_{\min},\ r_{\max}],\ \forall k$ (see Fig. 1). HETLORA comprises three steps: 1) Distribution via Truncation, 2) Local Training with Rank Self-Pruning, and 3) Sparsity-Weighted Aggregation of the LoRA modules. These steps are detailed further in the subsequent paragraphs. An overview of HETLORA is illustrated in Fig. 2.

1) Distribution via Truncation. At the begining of each communication round t, the server holds initial global LoRA modules $\overline{\mathbf{B}}^{(t)}$, $\overline{\mathbf{A}}^{(t)}$ with a global rank $r^{(t)}$. The value of the global rank $r^{(t)}$ depends on how we aggregate the heterogeneous rank LoRA modules which is elaborated on in step 3). The server then distributes these global LoRA modules to a subset of selected set of clients $\mathcal{S}^{(t)}$ with heterogeneous ranks $r_k^{(t)}, k \in \mathcal{S}^{(t)}$ for local training[1]. With the given global LoRA modules, we consider a simple and intuitive method of *truncation* where the server sends $\overline{\mathbf{B}}_{:,:r_k}^{(t)}$, $\overline{\mathbf{A}}_{:r_k,:}^{(t)}$ to each client k with rank $r_k^{(t)}$ for local training where we omitted the superscript for r_k for simplicity.

2) Local Training with Rank Self-Pruning.

[1]There is a superscript t for the ranks $r_k^{(t)}$ across clients which indicates that in HETLORA these heterogeneous ranks can be changed over the communication rounds via self-pruning explained in step 2).

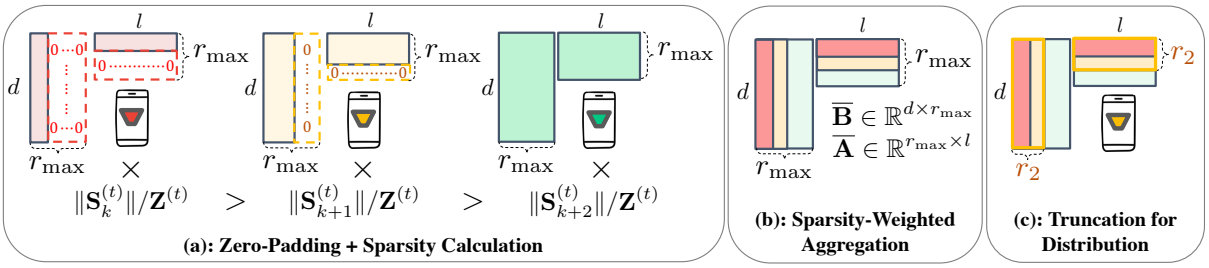


Figure 2: Overview of the zero-padding, sparsity-weighted aggregation, and truncation method for HETLORA; (a): Zero-pad LoRA modules with smaller ranks to $r_{\max}$ (clients with rank $r_{\max}$ does not need padding) and calculate their sparsity by calculating the Frobenius norm of the reconstructed model $\Delta\mathbf{W}_k^{(t)} = \mathbf{B}_k^{(t)}\mathbf{A}_k^{(t)}$; (b): After padding, aggregate all of the clients' LoRA modules with the weights $\|\mathbf{S}_k^{(t)}\|/\mathbf{Z}^{(t)}$ calculated by $\Delta\mathbf{W}_k^{(t)}$ to get the global LoRA modules; (c): Truncate the global LoRA modules for the specific rank of the next selected client (example for client with rank r_2).

After receiving LoRA modules from the server as $\mathbf{B}_k^{(t,0)} = \overline{\mathbf{B}}_{:,:r_k}^{(t)}$, $\mathbf{A}_k^{(t,0)} = \overline{\mathbf{A}}_{:r_k,:}^{(t)}$, each client $k \in \mathcal{S}^{(t)}$ performs τ local iterations of mini-batch SGD on their local data to minimize the local objective $\frac{1}{|\mathcal{B}_k|}\sum_{\xi\in\mathcal{B}_k}\ell((\mathbf{B}_k,\ \mathbf{A}_k),\xi|\mathbf{W}_0)$, and sends back the updated LoRA modules $\mathbf{B}_k^{(t,\tau)} \in \mathbb{R}^{d\times r_k^{(t)}}$ and $\mathbf{A}_k^{(t,\tau)} \in \mathbb{R}^{r_k^{(t)}\times l}$ to the server. This is the same process as the standard local training step in vanilla FedAvg (McMahan et al., 2017). However, we improve this vanilla local training step by adding a rank self-pruning mechanism where clients self-prune their respective ranks depending on the magnitude of the model parameters.

Specifically, we add a regularization term to the original local objective to get $\min_{\mathbf{B}_k,\,\mathbf{A}_k}\frac{1}{|\mathcal{B}_k|}\sum_{\xi\in\mathcal{B}_k}\ell((\mathbf{B}_k,\quad\mathbf{A}_k),\xi|\mathbf{W}_0) + \lambda\,\|\mathbf{B}_{k,:,r_k\gamma:r_k}\|\,\|\mathbf{A}_{k,r_k\gamma:r_k,:}\|$ where $\gamma < 1$ is a decay-factor that determines how aggressively we want to prune the ranks to a smaller value. The regularization term aims to minimize the norm of the last few ranks, which will become smaller if the first loss term $\frac{1}{|\mathcal{B}_k|}\sum_{\xi\in\mathcal{B}_k}\ell((\mathbf{B}_k,\ \mathbf{A}_k),\xi|\mathbf{W}_0)$ is not very large. After training with the new local objective we compare the norm of the updated LoRA modules' last few layers $\|\mathbf{B}_{k,:,r_k\gamma:r_k}\|\,\|\mathbf{A}_{k,r_k\gamma:r_k,:}\|$ with the ones from the initially received LoRA modules. If the former is smaller we prune the last few layers (pruning intensity is determined by γ) and send back the LoRA modules with a smaller rank. This means that for the LoRA modules which incurs a small local loss, i.e., well-trained on the clients' local data, the LoRA modules are more likely to be pruned to a smaller rank.

Such pruning allows HETLORA to reduce the noise in the LoRA modules introduced by clients having a larger rank than the actual rank that their data complexity requires, and also reduces the complexity of the LoRA modules to improve generalization and prevent overfitting (see Table 4). Once the rank is pruned for a client, the client saves the updated rank and uses it as the starting rank if selected for future communication rounds. The client then sends back their updated and possibly rank-pruned LoRA modules to the server to be processed in the aggregation step.

3) Sparsity-Weighted Aggregation. Finally, the last step of HETLORA is aggregating the received heterogeneous LoRA modules $\mathbf{B}_k^{(t,\tau)}$, $\mathbf{A}_k^{(t,\tau)}, k \in \mathcal{S}^{(t)}$. A straightforward way to aggregate the hetergeneous LoRA modules is using *zero-padding* to all the received LoRA modules with $r_i^{(t)} < \max\{r_k^{(t)}|k \in \mathcal{S}^{(t)}\}$ and then perform simple averaging over the modules. However, such naive aggregation can lead to biasing the model towards higher rank clients even when these clients may not hold valuable training information, i.e., having low data complexity, giving noisy updates.

In an ideal scenario where we can deploy any rank to any client, deploying higher ranks to the clients with higher data complexity or larger local datasets can retrieve more informative and less sparse updates from the clients. Conversely if we assign higher ranks to the clients whose data complexity is low, the actual rank of the full model from the reconstructed LoRA modules can be smaller than the assigned rank. Thus the higher rank client's update may be unnecessarily overemphasized in the naive zero padding method.

Based on this insight we propose a sparsity-

weighted aggregation scheme where the server reconstructs these LoRA modules to the full model as $\Delta \mathbf{W}_k^{(t)} = \mathbf{B}_k^{(t)} \mathbf{A}_k^{(t)}$ and gets the norm of the singular value vectors from the full models denoted as $\mathbf{S}_k^{(t)}$ by calculating $\|\Delta \mathbf{W}_k^{(t)}\|_F$. Note that the costly process of performing SVD for each of the full model $\Delta \mathbf{W}_k^{(t)}$ can be avoided by simply calculating the Frobenius norm of $\Delta \mathbf{W}_k^{(t)}$ (see Lemma 1.2 in (Guruswami and Kannan, 2012)). The server then weighs the LoRA modules with aggregation weight $p_k^{(t)}$ which is proportional to the norm of the singular value vectors. Formally, we have the the global LoRA modules updated as $\overline{\mathbf{B}}^{(t+1)} = \sum_{k \in \mathcal{S}^{(t)}} p_k^{(t)} \mathbf{B}_k^{(t)}$, $\overline{\mathbf{A}}^{(t+1)} = \sum_{k \in \mathcal{S}^{(t)}} p_k^{(t)} \mathbf{A}_k^{(t)}$ where $p_k^{(t)} := \|\mathbf{S}_k^{(t)}\| / \mathbf{Z}^{(t)}$ with normalizing factor $\mathbf{Z}^{(t)} := \sum_{k' \in \mathcal{S}^{(t)}} \|\mathbf{S}_{k'}^{(t)}\|$. This way, we can de-emphasize the larger rank assigned-clients that have rather less informative updates, and more emphasize the smaller rank assigned-clients that have more informative ones.

3.4 Why not Simply Reconstruct First, then Redistribute the LoRA modules?

One might ask why not simply reconstruct each of the LoRA modules to the full matrix and aggregate them. Here we show that reconstructing the LoRA modules and aggregating them to get the full model results in a different full model compared to when we aggregate the LoRA modules first and then reconstruct the final model. In Section 4 we also empirically show that reconstructing the LoRA modules to the full model and redistributing them after truncated SVD to the corresponding rank of the clients results in an underwhelming performance compared to HETLORA.

Let us consider a simple case where there are 2 clients with heterogeneous rank lora modules $\mathbf{B}_1 \in \mathbb{R}^{d \times 1}$, $\mathbf{A}_1 \in \mathbb{R}^{1 \times l}$ and $\mathbf{B}_2 \in \mathbb{R}^{d \times 1}$, $\mathbf{A}_2 \in \mathbb{R}^{2 \times l}$ respectively for client 1 and client 2 where the former has rank 1 and latter has rank 2. We set the notation for the LoRA modules' i^{th} row and j^{th} column value for $\mathbf{B}_k$ and $\mathbf{A}_k$ as $b_{k,ij}$ and $a_{k,ij}$ respectively. Then with $d = 3$, $l = 2$, when we reconstruct each of the LoRA modules first and then aggregate the full model we have its i^{th} row and j^{th} column as $(\sum_{k=1}^{2} b_{k,i0} a_{k,0j}) + b_{2,i1} a_{2,1j}$ and aggregating the LoRA modules first and then reconstructing the model has the full model's i^{th} row and j^{th} column as $(\sum_{k=1}^{2} b_{k,i0})(\sum_{k=1}^{2} a_{k,0j}) + b_{2,i1} a_{2,1j}$.

One can observe that the difference between the two models are the cross-terms between the left and right module of different client 1 and 2, i.e., $b_{1,i0} a_{2,0j} + b_{2,i0} a_{1,0j}$ for the i^{th} row and j^{th} column. In other words, when we reconstruct the LoRA modules first and then aggregate them to get the full model, each term in the full model are cross-products between the left and right module of each client and not the cross-products between clients. Thus, reconstructing the LoRA modules loses information on the cross-relation across clients, only retaining the knowledge on the cross-relation between the LoRA modules $\mathbf{B}$ and $\mathbf{A}$. Such observation is also corroborated by the *reconstruction first*'s underwhelming performance in Table 3.

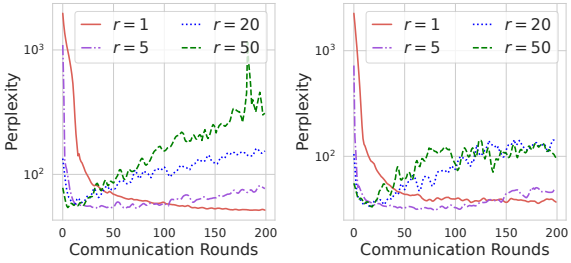
4 Experiments

In this section, we show that our proposed HETLORA outperforms its baselines in terms of **training speed, computation/communication efficiency, and final achieved performance**. We first show the performance of homogeneous LoRA to show how LoRA in general performs for low and high rank values. Second, as our main result, we demonstrate HETLORA's performance for different r_{min} and r_{max} values comparing them with full fine-tuning, homogeneous LoRA, and the reconstruction-first method elaborated in Section 3.4. We also conduct an ablation study on HETLORA with varying decay factor γ for the rank self-pruning step.

Settings. We use the transformer-based language model PaLM 2 (Google, 2023) of size XXS and XS for our experiments which are lightweight enough to be ODFMs (Google DeepMind, 2023) compared to standard FMs. The tasks we consider are the chat dialogue from the multi-session chat (MSC) dataset (Xu et al., 2021) and the text summarization task from the Reddit dataset (Völske et al., 2017). For the MSC data we use perplexity (Zhang et al., 2018) as the metric which has been used to show the quality of chat responses from generative models from previous literature (Sedoc et al., 2019). We sample 100 users uniformly at random and partition their data for training and evaluation by each `previous_dialogs` and `dialog`. The Reddit text summarization data consists of real users' reddit posts and their summarization, and

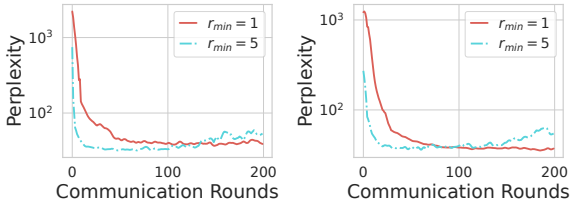
	Reddit (RougeL)		Multi-Session Chat (Perplexity)	
	PaLM 2-XXS	PaLM 2-XS	PaLM 2-XXS	PaLM 2-XS
Full	**94.56**($\pm$0.01)	**94.87**($\pm$0.04)	**32.70**($\pm$0.17)	**23.40**($\pm$0.36)
HomLoRA $r = 5$	92.57($\pm$1.56), $\times$0.001	92.89($\pm$0.96)	80.51($\pm$8.32), $\times$0.001	64.59($\pm$9.31)
HomLoRA $r = 50$	70.57($\pm$2.13), $\times$0.01	84.95($\pm$1.59)	307.96($\pm$11.43), $\times$0.01	167.46($\pm$1.72)
Recon+SVD	63.28($\pm$1.92), $\times$0.003	75.17($\pm$1.25)	323.89($\pm$20.57), $\times$0.002	215.63($\pm$15.38)
HetLoRA $\gamma = 0.99$	**94.23**($\pm$0.03), $\times$0.003	**94.41**($\pm$0.05)	**53.93**($\pm$1.57), $\times$0.002	**38.76**($\pm$0.52)

Table 3: Final RougeL score for Reddit text summarization and perplexity for multi-session chat for different federated fine-tuning methods. The blue text indicates the ratio of trained number of parameters compared to the full fine-tuning case. HetLoRA outperforms both HomLoRA and Recon+SVD method, but slightly underperforms the full fine-tuning case. However, compared to full fine-tuning the number of trained parameter is significantly smaller.



(a) PaLM 2-XXS **(b)** PaLM 2-XS

Figure 3: Performance of homogeneous LoRA for different rank r. Higher ranks achieve better performance with fewer communication rounds than the lower ranks, but they overfit quickly. Conversely, the lowest rank $r = 1$ achieves low perplexity slower than higher ranks, but without overfitting.



(a) $r_{\max} = 10$ **(b)** $r_{\max} = 50$

Figure 4: Performance of HetLoRA without rank pruning or and with simple average aggregation. Similar to homogeneous LoRA, larger $r_{\min}$ leads to overfitting for heterogeneous LoRA, but it is not as severe as homogeneous LoRA even for larger maximum rank $r_{\max} = 50$ showing that the smaller rank LoRA modules act as a regularizer for HetLoRA.

we use RougeL (Lin, 2004) as the metric. We use 298 users from Reddit that have at least 100 data samples as the training clients and use another 100 users with at least 100 data samples for evaluation. We use mini-batch size 8 and number of local iterations $\tau = 5$ with the feature length set to 1024. For the learning rate we perform grid search in $\eta = \{0.1,\ 0.01,\ 0.001,\ 0.0001\}$. For each MSC and Reddit task, we select 5 and 10 clients per communication round respectively. The rank distribution across clients for HetLoRA is set to a truncated power-law distribution with $\alpha = 0.1$ in the range between $[r_{\min}, r_{\max}]$ (inclusively). All experiments were ran with 3 different random seeds.

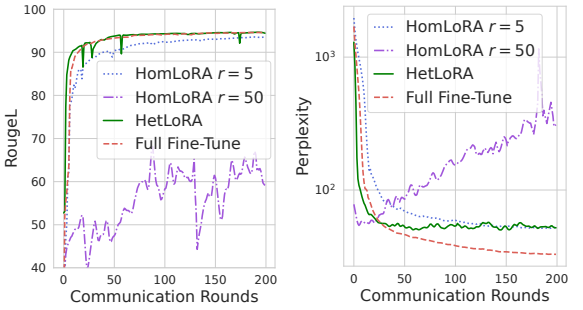
4.1 Experiment Results

Homogeneous LoRA and the Effect of Ranks r. First, we evaluate the performance of federated fine-tuning of the LoRA modules with homogeneous LoRA deployment across clients in Fig. 3 for different ranks $r \in [1, 5, 20, 50]$. We observe that a higher rank r for homogeneous LoRA achieves better perplexity floor with fewer communication rounds than the lower ranks but quickly overfits resulting in worse performance compared to the lower ranks after more communication rounds. On the other hand, while the lower rank cases need more communication rounds to achieve good performance, it does not have the problem of overfitting as the higher ranks. Hence for homogeneous LoRA, there is a trade-off between low and high ranks, in terms of faster performance achievement and overfitting. Note that these observations are consistent with previous literature in the centralized setting where a higher rank does not necessarily yields the best performance (Hu et al., 2021; Zhang et al., 2023b). Subsequently, we show that HetLoRA that is not only more practical than homogeneous LoRA, but also achieves a better final performance with fewer number of communication rounds without the overfitting issue.

Naïve Heterogeneous LoRA and the Effect of $r_{\min}$ and $r_{\max}$. First, we show the performance of *naïve* heterogeneous LoRA which is HetLoRA *without* self rank-pruning and *with only* average aggregation instead of the sparsity-weighted aggregation in Fig. 4. We see that similar to homogeneous LoRA, a smaller minimum rank $r_{\min} = 1$ leads to slower training but better performance while a larger maximum rank leads to faster training but worse performance. However, unlike homogeneous LoRA, the overfitting does not get

	Reddit (RougeL)		Multi-Session Chat (Perplexity)	
	PaLM 2-XXS	PaLM 2-XS	PaLM 2-XXS	PaLM 2-XS
HETLORA, $\gamma = 1$	92.17 ($\pm$0.08)	91.95 ($\pm$0.03)	55.07 ($\pm$0.81)	40.92 ($\pm$0.58)
HETLORA, $\gamma = 0.99$	**94.23** ($\pm$0.03)	**94.41** ($\pm$0.05)	**53.93** ($\pm$1.57)	**38.76** ($\pm$0.52)
HETLORA, $\gamma = 0.95$	89.62 ($\pm$1.33)	83.19 ($\pm$1.70)	71.10 ($\pm$1.39)	46.39 ($\pm$0.87)
HETLORA, $\gamma = 0.85$	60.31 ($\pm$3.04)	53.28 ($\pm$2.47)	120.72 ($\pm$10.93)	59.67 ($\pm$1.98)

Table 4: Ablation study on the effect of the decaying factor γ for HETLORA's self-rank pruning in the local training step. While aggressive pruning can be harmful to HETLORA's performance, pruning ($\gamma = 0.99$) can outperform the case when there is no pruning at all ($\gamma = 1$) by reducing the noise introduced by large rank clients with low data complexity.



(a) Reddit **(b)** MSC

Figure 5: Comparison of the performance across homogeneous LoRA, heterogeneous LoRA, and full fine-tuning. Heterogeneous LoRA achieves better performance than homogeneous LoRA with fewer number of communication rounds.

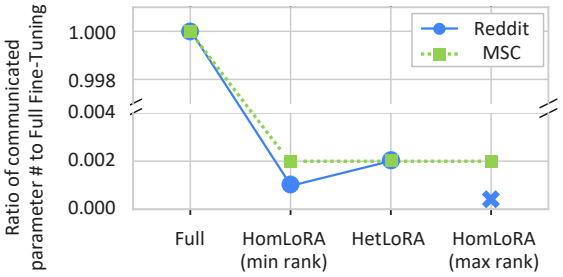


Figure 6: Ratio of communicated number of parameters for different PEFT methods to full fine-tuning to achieve the target value for the metric where it is RougeL 80 for Reddit text summarization task and perplexity 150 for the multi-session chat response task. The 'X' means that the target metric is not achieved even after convergence.

as severe for heterogeneous LoRA even with much larger ranks such as $r_{\text{max}} = 50$. We can imply from this result that the smaller rank LoRA modules act as a regularizer in heterogeneous LoRA preventing overfitting. Next, we show that by adding the self rank-pruning and sparsity-weighted aggregation, we improve the performance such that even with $r_{\text{min}} = 5$ we are able to prevent overfitting issues and achieve better training speed and final performance than other baselines.

Heterogeneous LoRA compared to PEFT. Finally, we compare HETLORA with other baselines in Table 3 and Fig. 5. We see that HETLORA with $r_{\text{min}} = 5$ and $r_{\text{max}} = 50$ achieves faster training as well as better performance than homogeneous LoRA cases with both edge cases of the ranks $r \in \{5, 50\}$ and reconostruction+SVD which was explained in Section 3.4. This implies that HETLORA is not only practical in the sense that clients are allowed to have their own rank values, it can also outperform the limited case of homogeneous LoRA where all clients have $r = r_{\text{min}}$ or the impractical case where all clients have $r = r_{\text{max}}$.

Heterogeneous LoRA compared to Full-Fine Tuning. In Table 3, we can see that for the Reddit text summarization, HETLORA achieves similar performance with full-fine tuning with only 0.003% of parameters trained compared to full-fine tuning. For chat response, HETLORA achieves slightly lower performance than full fine-tuning. However, full fine-tuning requires to train a much larger number of parameters compared to HETLORA, making it infeasible to train with ODFMs in practice. We also show in Fig. 6 that to achieve the targeted performance for both Reddit and MSC task, HETLORA requires significantly less number of parameters to be trained and communicated compared to full fine-tuning. Although for Reddit, HOMLORA has a slightly less number of parameters to be trained, the final achieved RougeL is outperformed by HETLORA as shown in Table 3.

Effect of the Decaying Factor γ. Lastly, we conduct an ablation study on the effect of the decaying factor γ of HETLORA's local training step with self-rank pruning in Table 4. We observed that aggressive pruning hurts the performance where $\gamma = 0.85$ shows the worse performance across the varying γ values. On the other hand, no pruning at all ($\gamma = 1$) underperforms the case when there is pruning ($\gamma = 0.99$), showing that reducing the noise introduced by large rank clients which data complexity is actually not that high indeed improves the performance.

5 Discussions and Concluding Remarks

In our work, we investigated federated fine-tuning for ODFMs by proposing HETLORA that enables utilizing the power of FMs at the edge by taking device system and data heterogeneity into consideration. HETLORA is not only practical but also achieves better training speed, communication/computation efficiency, and final performance compared to other relevant baselines such as SVD-based LoRA, homogeneous LoRA, or full fine-tuning. Our findings in this work opens up several questions worth investigating. For instance, if the settings allow us to assign specific ranks to clients what will be the effective way to assign the ranks across clients for better convergence and performance? Another important next step of our work includes pursuing the theoretical convergence and generalization of heterogeneous LoRA.

6 Limitations

In this work, we address tackling system and data heterogeneity in federated fine-tuning of on-device foundation models. Our work is motivated by clients being able to carry different ranks for the LoRA fine-tuning method depending on their available resources, and thus exploiting this characteristic to improve federated fine-tuning with heterogeneous LoRA. However, our work assumes that the rank distribution across clients (which is analogous to how system resources are distributed across clients) is independent to the data distribution. There can be scenarios in which this is not necessarily the case where the rank and data distribution can be correlated. For instance, more affluent populations can have better off devices with larger resource capacity, and may have data distributions different to that of less affluent populations. Such correlation should be explored for future work to better understand the implications of heterogenoeus LoRA.

References

Anonymous. 2024. Flora: Federated fine-tuning large language models with heterogeneous low-rank adaptations.

Sara Babakniya, Ahmed Roushdy Elkordy, Yahya H. Ezzeldin, Qingfeng Liu, Kee-Bong Song, Mostafa El-Khamy, and Salman Avestimehr. 2023. Slora: Federated parameter efficient fine-tuning of language models. *CoRR*, abs/2308.06522.

Jiamu Bai, Daoyuan Chen, Bingchen Qian, Liuyi Yao, and Yaliang Li. 2024. Federated fine-tuning of large language models under heterogeneous tasks and client resources. *arXiv preprint arXiv:2402.11505*.

Rishi Bommasani, Drew A. Hudson, and et. al. Ehsan Adeli. 2022. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*.

Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. 2016. Practical secure aggregation for federated learning on user-held data. In *NIPS Workshop on Private Multi-Party Machine Learning*.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Zhiqi Bu, Yu-Xiang Wang, Sheng Zha, and George Karypis. 2022. Differentially private bias-term only fine-tuning of foundation models. *arXiv preprint arXiv:2210.00036*.

Jinyu Chen, Wenchao Xu, Song Guo, Junxiao Wang, Jie Zhang, and Haozhao Wang. 2022. Fedtune: A deep dive into efficient federated fine-tuning with pre-trained transformers. *CoRR*, abs/2211.08025.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Danny Driess, Fei Xia, Mehdi S. M. Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, Wenlong Huang, Yevgen Chebotar, Pierre Sermanet, Daniel Duckworth, Sergey Levine, Vincent Vanhoucke, Karol Hausman, Marc Toussaint, Klaus Greff, Andy Zeng, Igor Mordatch, and Pete Florence. 2023. Palm-e: An embodied multimodal language model. *arXiv preprint arXiv:2303.03378*.

Gemini Team Google. 2024. Gemini: A family of highly capable multimodal models. https://arxiv.org/abs/2312.11805.

Google. 2022. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*.

Google. 2023. Palm 2 technical report. *arXiv preprint arXiv:2305.1040*.

Google DeepMind. 2023. Introducing palm2. https://blog.google/technology/ai/google-palm-2-ai-large-language-model/.

Tao Guo, Song Guo, and Junxiao Wang. 2023. Pfedprompt: Learning personalized prompt for vision-language models in federated learning. In *Proceedings of the ACM Web Conference 2023*, WWW '23, page 1364–1374, New York, NY, USA. Association for Computing Machinery.

Tao Guo, Song Guo, Junxiao Wang, and Wenchao Xu. 2022. Promptfl: Let federated participants cooperatively learn prompts instead of models — federated learning in age of foundation model. *CoRR*, abs/2208.11625.

Venkatesan Guruswami and Ravi Kannan. 2012. Lecture notes in computer science theory for the information age.

Samuel Horváth, Stefanos Laskaridis, Shashank Rajput, and Hongyi Wang. 2024. Maestro: Uncovering low-rank structures via trainable decomposition. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Neil Houlsby, Andrei Giurgiu, Stanisław Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. In *The 36th Conference on Neural Information Processing Systems (NeurIPS 2022)*.

Jaejun Lee, Raphael Tang, and Jimmy Lin. 2019. What would elsa do? freezing layers during transformer fine-tuning. *arXiv preprint arXiv:1911.03090*.

Gwen Legate, Nicolas Bernier, Lucas Caccia, Edouard Oyallon, and Eugene Belilovsky. 2023. Guiding the last layer in federated learning with pre-trained models. In *Workshop of Federated Learning and Analytics in Practice@ICML*.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Zequan Liu, Jiawen Lyn, Wei Zhu, Xing Tian, and Yvette Graham. 2024. Alora: Allocating low-rank adaptation for fine-tuning large language models. *arXiv preprint arXiv:2403.16187*.

Andre Manoel, Mirian del Carmen Hipolito Garcia, Tal Baumel, Shize Su, Jialei Chen, Robert Sim, Dan Miller, Danny Karmon, and Dimitrios Dimitriadis. 2023. Federated multilingual models for medical transcript analysis. In *Conference on Health, Inference, and Learning (CHIL)*, pages 147–162.

H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agøura y Arcas. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. *International Conference on Artificial Intelligenece and Statistics (AISTATS)*.

OpenAI. 2023. Gpt-4 technical report. *arXiv preprint arXiv:submit/4812508*.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. 2021. Learning transferable visual models from natural language supervision. *arXiv preprint arXiv:2103.00020*.

Anit Kumar Sahu, Tian Li, Maziar Sanjabi, Manzil Zaheer, Ameet Talwalkar, and Virginia Smith. 2020. Federated optimization for heterogeneous networks. In *Proceedings of the 3rd MLSys Conference*.

Joao Sedoc, Daphne Ippolito, Arun Kirubarajan, Jai Thirani, Lyle Ungar, and Chris Callison-Burch. 2019. Chateval: A tool for chatbot evaluation. *Proceedings of NAACL-HLT*.

Ofir Ben Shoham and Nadav Rappoport. 2023. Federated learning of medical concepts embedding using behrt. *arXiv preprint arXiv:2305.13052*.

Aliaksandra Shysheya, John F Bronskill, Massimiliano Patacchiola, Sebastian Nowozin, and Richard E Turner. 2023. Fit: Parameter efficient few-shot transfer learning for personalized and federated image classification. *International Conference on Learning Representations (ICLR)*.

Youbang Sun, Zitao Li, Yaliang Li, and Bolin Ding. 2024. Improving lora in privacy-preserving federated learning. In *International Conference on Learning Representations (ICLR)*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Michael Völske, Martin Potthast, Shahbaz Syed, and Benno Stein. 2017. TL;DR: Mining Reddit to learn automatic summarization. In *Proceedings of the Workshop on New Frontiers in Summarization*, pages 59–63, Copenhagen, Denmark. Association for Computational Linguistics.

Shiqiang Wang, Tiffany Tuor, Theodoros Salonidis, Kin K. Leung, Christian Makaya, Ting He, and Kevin Chan. 2019. Adaptive federated learning in resource constrained edge computing systems. *IEEE Journal on Selected Areas in Communications*, 37(6):1205 – 1221.

Colin Wei, Sang Michael Xie, and Tengyu Ma. 2021. Why do pretrained language models help in downstream tasks? an analysis of head and prompt tuning. *Advances in Neural Information Processing Systems*, 34:16158–16170.

Mitchell Wortsman, Suchin Gururangan, Shen Li, Ali Farhadi, Ludwig Schmidt, Michael Rabbat, and Ari S. Morcos. 2023. lo-fi: distributed fine-tuning without communication. *Transactions on Machine Learning Research (TMLR)*.

Xinghao Wu, Xuefeng Liu, Jianwei Niu, Haolin Wang, Shaojie Tang, and Guogang Zhu. 2024. Fedlora: When personalized federated learning meets low-rank adaptation.

Jing Xu, Arthur Szlam, and Jason Weston. 2021. Beyond goldfish memory: Long-term open-domain conversation. *arXiv preprint arXiv:2107.07567*.

Liping Yi, Han Yu, Gang Wang, and Xiaoguang Liu. 2023. Fedlora: Model-heterogeneous personalized federated learning with lora tuning. *arXiv preprint arXiv:2310.13283*.

Sixing Yu, J. Pablo Muñoz, and Ali Jannesari. 2023. Federated foundation models: Privacy-preserving and collaborative learning for large models. *arXiv preprint arXiv:2305.11414*.

Elad Ben Zaken, Shauli Ravfogel, and Yoav Goldberg. 2021. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. *arXiv preprint arXiv:2106.10199*.

Jianyi Zhang, Saeed Vahidian, Martin Kuo, Chunyuan Li, Ruiyi Zhang, and Guoyin Wangand Yiran Chen. 2023a. Towards building the federated gpt: Federated instruction tuning. *CoRR*, abs/2305.05644.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. 2023b. Adaptive budget allocation for parameter-efficient fine-tuning. In *The 11th International Conference on Learning Representations (ICLR)*.

Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. 2018. Personalizing dialogue agents: I have a dog, do you have pets too? *In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Zhuo Zhang, Xiangjing Hu, Jingyuan Zhang, Yating Zhang, Hui Wang, Lizhen Qu, and Zenglin Xu. 2023c. Fedlegal: The first real-world federated learning benchmark for legal nlp. *In Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL)*.

Zhuo Zhang, Yuanhang Yang, Yong Dai, Qifan Wang, Yue Yu, Lizhen Qu, and Zenglin Xu. 2023d. Fedpetuning: When federated learning meets the parameter-efficient tuning methods of pre-trained language models. *Findings of the Association for Computational Linguistics (ACL)*.

Ce Zhou, Qian Li, Chen Li, Jun Yu, Yixin Liu, Guangjing Wang, Kai Zhang, Cheng Ji, Qiben Yan, Lifang He, Hao Peng, Jianxin Li, Jia Wu, Ziwei Liu, Pengtao Xie, Caiming Xiong, Jian Pei, Philip S. Yu, and Lichao Sun. 2023. A comprehensive survey on pretrained foundation models: A history from bert to chatgpt. *arXiv preprint arXiv:2302.09419*.