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

Yae Jee Cho
Carnegie Mellon University
yaejeec@andrew.cmu.edu

Luyang Liu
Google Research
luyangliu@google.com

Zheng Xu
Google Research
xuzheng@google.com

Aldi Fahrezi
Google Research
aldifahrezi@google.com

Matt Barnes
Google Research
mattbarnes@google.com

Gauri Joshi
Carnegie Mellon University
gaurij@andrew.cmu.edu

Abstract

Foundation models (FMs) in massive parameter space pretrained on a large amount of (public) data perform remarkably well on various downstream tasks with just a few samples for fine-tuning. However, direct fine-tuning of the standard FMs often becomes difficult due to their massive size, especially for scenarios where FMs are adapted on private data distributed across resource-limited devices. As such, only those FMs with relatively small parameter size may be capable of on-device fine-tuning. We call these smaller FMs as *on-device FMs (ODFMs)*. In our work, we investigate parameter-efficient federated fine-tuning of ODFMs (XXS PaLM2) for downstream tasks on devices using low-rank approximations (LoRAs), where we investigate multi-session chat data from real clients as the downstream task of interest. We first examine federated fine-tuning with homogeneous LoRA ranks across clients, and show that higher ranks can lead to overfitting despite their faster learning speed whilst lower ranks do not overfit but converge slower in training. Based on these observations, we propose heterogeneous LoRA, where we deploy *hetergeneous ranks* across clients, aggregate the heterogeneous LoRA modules through zero-padding, and redistribute the LoRA modules heterogeneously through truncation. Our proposed heterogeneous LoRA is simple yet effective. It achieves the best of both worlds by combining the advantages of high-rank and low-rank LoRAs. This allows us to achieve the best performance with the fewest number of communication rounds, while also avoiding the problem of overfitting.

1 Introduction

The emerging foundation models (FMs) [1, 2, 3, 4, 5, 6] have shown surprising zero/few shot learning capabilities, performing well on a variety of tasks including text/image generation with prompts, language translation, solving math problems, and conversing in natural language. FMs such as the GPT [5, 7] or PaLM series [6, 8, 9] are trained on a massive, variety of data (mostly unlabeled) with parameters ranging up to hundreds of billions at size, making it capable to be applied to different domains with just a few additional training rounds (i.e., fine-tuning) on the targeted dataset. The colossal size of these FMs, however, requires large number of resources for directly fine-tuning their entire parameter space. To tackle this issue, many recent works have proposed different parameter-efficient fine-tuning (PEFT) methods of FMs such as prompt tuning [10], utilizing adapters [11], or low-rank adaptation (LoRA) of the original model [12] which freezes the original pre-trained parameters of the FM and train additional, smaller number of parameters instead.

The aforementioned work on PEFT methods for FMs, however, assume that i) the pre-trained FMs are deployed to and trained with the data of a *single* machine/client for adaptation to the downstream task and that ii) the client has enough resources to even fit a standard FM of hundred billion size for,

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

at least, inference. In practice, there are often cases where we are interested in fine-tuning FMs for on-device private data that is distributed across multiple devices (clients). For instance, sensitive and private data such as medical information or law-related documents may be hard to collect centrally in a private manner and fine-tuning of the FMs may need to be done at the edge [13, 14, 15]. We define *on-device FMs (ODFMs)* as models with fewer than several billion parameters that are able to fit into memory on limited capacity clients considering current hardwares. We also use the term *federated fine-tuning* which we define as training a set of parameters collaboratively 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) [16], which aims to train parameters that perform well for individual clients rather than downstream tasks.

	Zero-Shot	Few-Shot	Full-Training
PaLM2 XXS	2930.23	2541.86	**23.71**
PaLM2 XS	2712.86	481.95	**18.32**

Table 1: Perplexity performance of PaLM2 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 details of experiments are in Section 3.)

Federated fine-tuning of ODFMs entails unique challenges that are not present in either standard PEFT of FMs or the standard federated training of models that are not FMs. First, unlike conventional task-specific models, FMs have their zero/few-shot learning capability often supported by their large parameter space that is trained on massive data. As shown in Table 1 and previous literature [17], however, FMs' performance can deteriorate as their sizes get smaller and federated fine-tuning may not merely be useful but *even inevitable for ODFMs* to perform well for downstream tasks on devices. Moreover, since devices usually have limited and heterogeneous system capabilities [18], PEFT methods that can flexibly adapt to the heterogeneous devices should be considered for federated fine-tuning of ODFMs. Previous work evaluated PEFT with FL via performing a general evaluation over different PEFT methods naïvely combined with FL [19, 20, 21, 22, 23]. However, these works does not investigate a more practical federated fine-tuning setting for ODFMs where PEFT methods are catered to heterogeneous system-capabilities such as by applying different PEFT strategies to different clients for performance improvement. In practice, it may even be difficult to choose LoRA rank parameters for clients, particularly for resource limited mobile devices with natural system heterogeneity.

In this work, we propose heterogeneous LoRA for federated fine-tuning to cater to system heterogeneity and outperform the naïve combination of LoRA and federated fine-tuning where homoegneous ranks are applied across clients. We show the performance of PaLM 2 [9] of XXS size for chat responses on the multi-session chat data [24] of real clients. Our contributions can be summarized as:

- 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 propose heterogeneous LoRA that can apply different rank LoRA modules to different clients via utilizing zero-padding and truncation for the aggregation and distribution of the heterogeneous size LoRA modules.
- We show that our proposed simple approach of heterogeneous LoRA outperforms naïvely applying homogeneous ranks across clients in terms of both training speed and final performance, gaining the best of both worlds of LoRA with high and low ranks.
- We evaluate how different distributions of the ranks across clients, specifically uniform and power-law distribution, affects the performance of heterogeneous LoRA.

2 Federated Fine-Tuning with LoRA

Formally, we define the pre-trained ODFM weights 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 [12], 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 decompositions of $\Delta\mathbf{W}$ with identical rank r in the centralized setting. Now, let us consider LoRA in the federated fine-tuning scenario where there are M total clients. Each client $k \in [M]$ has its local training dataset $\mathcal{B}_k$ 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 value 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

$\overline{\mathbf{W}} = \mathbf{W}_0 + \overline{\mathbf{B}}\,\overline{\mathbf{A}}$. Throughout the paper, we denote the truncation of a matrix with the : symbol. For instance, if there is no truncation at the row but truncation from $x < l$ at the column for the matrix $\mathbf{W} \in \mathbb{R}^{d \times l}$, we keep all the columns until x and omit the last $l - x$ columns and denote the resulting matrix it as $\mathbf{W}[:, : x]$.

2.1 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 [25]. 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 finally 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 send back to the next set of selected clients for the next communication round. This training process is nearly identical to the standard FL algorithm [25] 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 clients, however, for federated fine-tuning it is possible and perhaps even more reasonable to consider heterogeneous rank deployment where each client trains different rank LoRA modules. Such setting can be motivated from inevitable system constraints of the clients [26] or simply to improve the federated fine-tuning performance which we show is possible in our experiments in Section 3. However, with heterogeneous ranks set for clients (Heterogeneous LoRA), it is non-trivial how to aggregate these modules at the server and then distribute the modules back to the next set of selected clients. Hence, we explore the research question of *"Can we aggregate the heterogeneous LoRA ranks r such that it performs better than homogeneous deployment of LoRA ranks with federated fine-tuning? How does the distribution of heterogeneous LoRA ranks across clients affect the performance of federated fine-tuning with heterogeneous LoRA?"* Towards this goal, we propose and explore a simple method for heterogeneous LoRA rank aggregation which we elaborate in the subsequent section, and show its performance compared to homogeneous LoRA and full fine tuning, giving insights into how different ranks in heterogeneous LoRA can affect the performance of federated fine-tuning.

2.2 Heterogeneous LoRA: Truncation and Padding

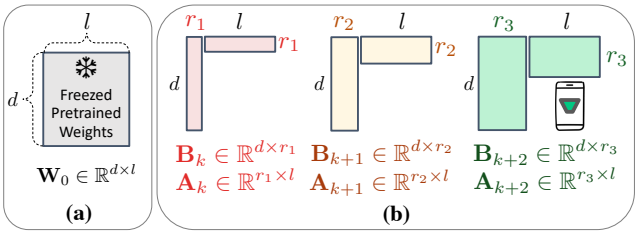


Figure 1: Example of (a): the pretrained weights $\mathbf{W}_0$ that are stored in the clients' devices and (b): the heterogeneous ranks assigned to different clients with $r_{\text{min}} = r_1 < r_2 < r_3 = r_{\text{max}}$.

In heterogeneous LoRA, we consider that different clients use different ranks to match their computation resources, and the ranks are assigned prior to training in simulation. We do not assume any specific function for assigning the ranks to the clients, and explore how different distributions of ranks (e.g., uniform, power-law) across clients affect the performance of heterogeneous LoRA in Section 3. Specifically, each client k has a rank r_k that is fixed throughout training where the ranks range from a minimum and maximum rank, i.e., $r_k \in [r_{\text{min}},\ r_{\text{max}}],\ \forall k$ (see Fig. 1). For heterogeneous LoRA, first, the server has to distribute the global LoRA modules with the global rank r to a subset of clients with heterogeneous ranks. How we set the global rank r will also vary depending on how we distribute and aggregate the heterogeneous rank LoRA modules.

In our work, we consider a simple and intuitive method of *truncation* where we set the global rank $r = r_{\text{max}}$ and for each client with rank r_k the server sends $\overline{\mathbf{B}}_k^{(t)}[:, : r_k],\ \overline{\mathbf{A}}_k^{(t)}[: r_k, :]$ for local training (see Fig. 2(c)). Then the client performs τ local iterations of mini-batch SGD on their local data to send back the updated LoRA modules $\mathbf{B}_k^{(t)} \in \mathbb{R}^{d \times r_k}$ and $\mathbf{A}_k^{(t)} \in \mathbb{R}^{r_k \times l}$. The next critical step to consider is how to aggregate the heterogeneous LoRA modules. Recall that

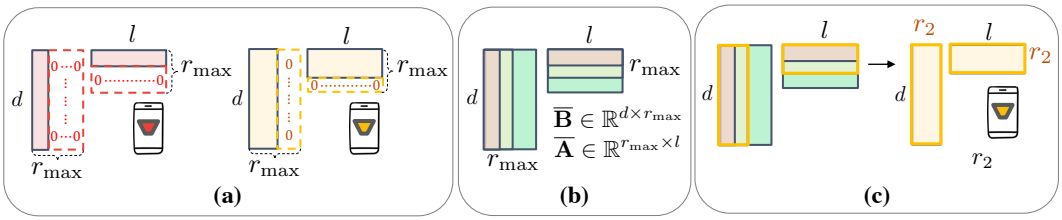


Figure 2: Overview of the zero-padding and truncation method for heterogeneous LoRA; (a): Zero-pad LoRA modules with smaller ranks to $r_{\max}$ (clients with rank $r_{\max}$ does not need padding); (b): After padding, aggregate all of the clients' LoRA modules to get the global LoRA modules; (c): Truncate the global LoRA modules for the specific rank of the next selected client (example for client with rank r_2).

	$r=1$	$r=5$	$r=10$	$r=20$	$r=50$	$r=100$	$r=150$	$r=200$
PaLM2 XXS	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 PALM2 XXS.

the final global LoRA module should have the rank $r = r_{\max}$. For this step we use simple *zero-padding* to all the received LoRA modules with $r_k < r_{\max}$ to the maximum rank $r_{\max}$. Formally, for the received LoRA modules at communication round t with $r_k < r_{\max}$, we extend the matrix from shape $\overline{\mathbf{B}}_k^{(t)} \in \mathbb{R}^{d\times r_k}$, $\overline{\mathbf{A}}_k^{(t)} \in \mathbb{R}^{r_k\times l}$ to the shape of $\overline{\mathbf{B}}_k^{(t)} \in \mathbb{R}^{d\times r_{\max}}$, $\overline{\mathbf{A}}_k^{(t)} \in \mathbb{R}^{r_{\max}\times l}$ where $\overline{\mathbf{B}}_k^{(t)}[:r_k : r_{\max}, : r_k : r_{\max}]$, $\overline{\mathbf{A}}_k^{(t)}[:r_k : r_{\max}, : r_k : r_{\max}]$ is set to 0. An illustration of the server-side truncation and aggregation via zero-padding is shown in Fig. 2 for the case when there are three heterogeneous rank LoRA modules r_1, r_2, r_3 where $r_{\max} = r_3$. Note that if we set all ranks to the same rank our truncation and padding method simply becomes analogous to the homogeneous LoRA case we explained in Section 2.1. Now that we have described our settings for both homogeneous and heterogeneous LoRA we show their performances on federated fine-tuning in the next section.

3 Experiments

3.1 Experiment Setup

We consider the transformer-based model PaLM2 [9] of size XXS which we believe are applicable to fit in to the category of ODFMs given their smaller sizes compared to standard FMs. The LoRA modules are applied to only the self-attention layers as proposed in the original LoRA paper [12] where their relative number of parameters compared to the original model are shown in Table 2. The task we consider in this paper is chat dialogue using the multi-session chat (MSC) dataset [24] which is a collection of human-human interactions comprising numerous extended chat sessions. For the metric, we use perplexity [27] which has been used to show the quality of chat responses from generative models from previous literature [28]. We sample 100 clients (real users) uniformly at random from the MSC dataset and partition their data for training and evaluation by each `previous_dialogs` and `dialog`. We fix the local mini-batch size to 8 and number of local iterations τ to 5 with the feature length set to 1024. The learning rates are sweeped across $0.01, 0.001, 0.0001$ to find the best performing one. The rank distribution across clients for heterogeneous LoRA, unless mentioned otherwise, is set to a uniform distribution between $[r_{\min}, r_{\max}]$ (inclusively).

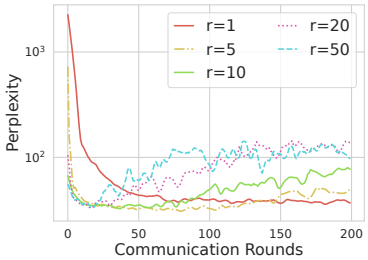
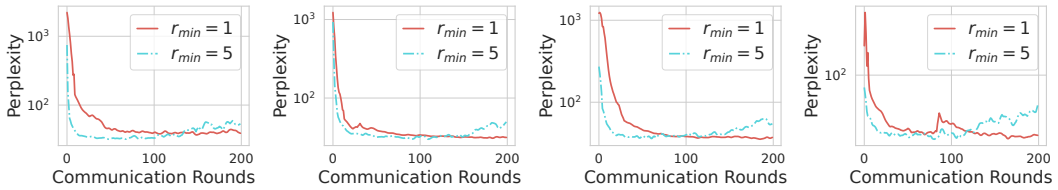


Figure 3: Performance of homogeneous LoRA for different rank r. Higher ranks achieve good performance faster than lower ranks, but it overfits while the lowest rank $r = 1$ achieves the same perplexity slower than higher ranks without overfitting.

3.2 Experiment Results

Homogeneous LoRA and the Effect of the 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 $r \in [1, 5, 10, 20, 50]$. We can observe a higher rank r for homogeneous LoRA

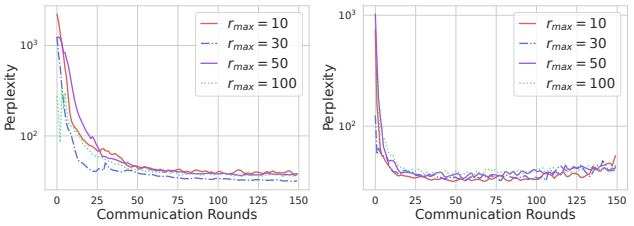


(a) $r_{\text{max}} = 10$ **(b)** $r_{\text{max}} = 30$ **(c)** $r_{\text{max}} = 50$ **(d)** $r_{\text{max}} = 150$

Figure 4: Performance of heterogeneous LoRA with different minimum rank values i.e., $r_{\text{min}} \in [1, 5]$. As similar to homogeneous LoRA, larger r_{min} leads to more overfitting for heterogeneous LoRA, but it is not as severe as the homogeneous LoRA even for larger maximum ranks $r_{\text{max}} = 150$ showing that the smaller rank LoRA modules act as a regularizer in heterogeneous LoRA.

achieves better perplexity floor with fewer communication rounds than lower rank cases but quickly overfits resulting in worse performance compared to the lower rank cases 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 rank cases. Hence for homogeneous LoRA, there is a trade-off to consider 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 [12, 29]. Next, in the subsequent paragraphs we show that with our proposed heterogeneous LoRA, we achieve good performance quickly without this overfitting issue, showing better performance than the homogeneous LoRA case.

Heterogeneous LoRA and the Effect of r_{min} and r_{max}. First, we show the performance of federated fine-tuning with heterogeneous LoRA for different r_{min} and r_{max} in Fig. 4 where we gain key observations that the minimum rank across clients largely affects the performance of federated fine-tuning. Specifically, a smaller minimum rank $r_{\text{min}} = 1$ leads to slower training but better performance while a larger maximum rank leads to faster training but worse performance, as similar to what we have seen in the homogeneous LoRA case. However, compared to homogeneous LoRA we can observe that the overfitting does not get as severe for heterogeneous LoRA even with much larger ranks such as $r_{\text{max}} = 150$. We can imply from this result that the smaller rank LoRA modules act as a regularizer in heterogeneous LoRA. Moreover, interestingly, we show in Fig. 5 that increasing the maximum rank r_{max} does not always improve the performance of federated fine-tuning. The best performing case that has fast training speed as well as good performance is in fact $r_{\text{max}} = 30, r_{\text{min}} = 1$ case in Fig. 5(a), where a higher r_{max} even with $r_{\text{min}} = 1$ leads to overfitting. For $r_{\text{min}} = 5$ in Fig. 5(b), the discrepancies across r_{max} are nearly negligible showing that the minimum rank r_{min} is critical to the performance of heterogeneous LoRA especially to prevent overfitting.



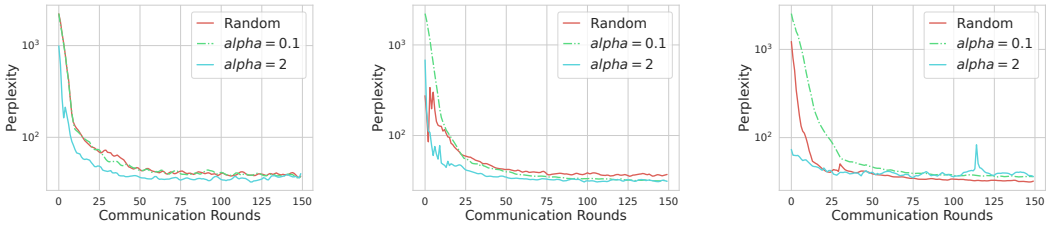
(a) $r_{\text{min}} = 1$ (b) $r_{\text{min}} = 5$

Figure 5: Performance of heterogeneous LoRA with varying r_{max}. The best performing case is $r_{\text{max}} = 30, r_{\text{min}} = 1$ showing that increasing the maximum rank does not always help. It is also observable that for $r_{\text{min}} = 5$ the discrepancies across maximum ranks are nearly negligible due to overfitting showing that the minimum rank r_{min} plays an important role in preventing overfitting.

Different Client Rank Distributions for Heterogeneous LoRA. So far we have only looked at uniformly distributed ranks across clients for the heterogeneous LoRA, but we further investigate differently distributed ranks with the power law distribution where a larger alpha indicates more clients are assigned with higher ranks and a smaller alpha indicates more clients are assigned smaller ranks (see Fig. 7). The mapping of a client to a rank is random as the same as the uniform distribution case. With the ranks distributed with either the power-law distribution ($\alpha \in [0.1,\ 2]$) or uniform distribution, we show in Fig. 8 that when clients' rank distribution is skewed towards lower ranks ($\alpha = 0.1$), although there is no overfitting, the training speed is slow. On the other hand, with clients' rank skewed towards higher ranks, the training speed is not only faster than uniform or $\alpha = 0.1$, but also is able to avoid overfittig for certain maximum rank values such as $r_{\text{max}} = 30$ shown in Fig. 8(b). This shows that for certain settings rank distributions to clients, heterogeneous LoRA can achieve the best of both worlds of lower and higher rank, where training speed is fast but also without

(a) Uniform Distribution **(b)** Power Distribution, $\alpha = 0.1$ **(c)** Power Distribution, $\alpha = 2$

Figure 7: Histogram of client ranks with $r_{\min} = 1,\ r_{\max} = 50$ for different distributions including uniform and power-law distribution for $\alpha \in [0.1, 2]$. A smaller α leads to clients having smaller ranks and vice versa.



(a) $r_{\max} = 10$ **(b)** $r_{\max} = 30$ **(c)** $r_{\max} = 50$

Figure 8: Performance of heterogeneous LoRA with different rank distributions (uniform (random), $\alpha \in [0.1, 2]$). More skew towards higher ranks ($\alpha = 2$) achieves better performance with fewer communication rounds compared to uniform (random) and $\alpha = 0.1$ for $r_{\max} \in [10, 30]$.

the overfitting issue most severely observed in homogeneous LoRA with high ranks. We directly compare homogeneous and heterogeneous LoRA in the next paragraph.

Heterogeneous LoRA compared to Homogeneous LoRA. Finally, we compare heterogeneous LoRA with the homogeneous LoRA deployment in Fig. 6. We see that our proposed heterogeneous LoRA with $r_{\min} = 1$ and $r_{\max} = 30$ achieves faster training as well as better performance than homogeneous LoRA cases with both edge cases of the ranks $r \in \{1, 30\}$. Hence going back to our main question of whether we can devise a aggregation/distribution scheme of heterogeneous LoRA for federated fine-tuning such that it out performs homoegneous LoRA, we show that this is indeed possible. However, we can also observe that heterogeneous LoRA is still not able to achieve better performance than full fine-tuning. Note that however, full fine-tuning requires to train a much larger number of parameters compared to heterogeneous LoRA per communication round (see Table 2). Nevertheless, this leaves room for further research questions of whether it is possible for heterogeneous LoRA to also outperform full federated fine tuning with the same number of communication rounds.

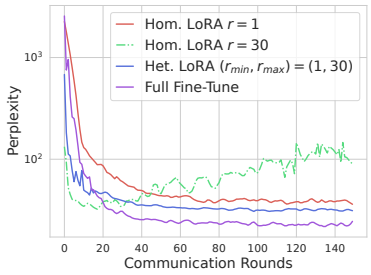


Figure 6: Comparison of the performance across homogeneous LoRA, heterogeneous LoRA, and full-fine tuning. Heterogeneous LoRA achieves better performance than homogeneous LoRA with fewer number of communication rounds. However, it is still not able to outperform full fine-tuning.

4 Discussions and Concluding Remarks

In our work, we investigate federated fine-tuning for ODFMs, specifically, utilizing heterogeneous LoRA. We show that heterogeneous LoRA, even with the simple approach of zero-padding and truncation, can achieve better training speed and final performance compared to homogeneous LoRA. We also show interesting findings consistent with previous literature [12, 29] that increasing ranks does not always help, even for heterogeneous LoRA. Our findings in this work opens up several questions worth investigating. First, we would like to investigate whether there is a way to improve our current approach of heterogeneous LoRA for federated fine-tuning to achieve comparable performance with full fine-tuning. The current method for heterogeneous LoRA is data agnostic with simple padding and truncation, but whether there is a way to incorporate awareness of data heterogeneity to heterogeneous LoRA is an interesting direction. Another open question is what is the right way to assign the ranks across clients. In our current version of heterogeneous LoRA we assign the ranks at random to the clients but whether we can assign the ranks such that heterogeneous LoRA performance can be improved remains open for investigation.

References

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

[2] 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. A comprehensive survey on pretrained foundation models: A history from bert to chatgpt. *arXiv preprint arXiv:2302.09419*, 2023.

[3] 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. Learning transferable visual models from natural language supervision. *arXiv preprint arXiv:2103.00020*, 2021.

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

[5] OpenAI. Gpt-4 technical report. *arXiv preprint arXiv:submit/4812508*, 2023.

[6] Google. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.

[7] 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. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

[8] 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. Palm-e: An embodied multimodal language model. *arXiv preprint arXiv:2303.03378*, 2023.

[9] Google. Palm 2 technical report. *arXiv preprint arXiv:2305.1040*, 2023.

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

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

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

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

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

[15] Zhuo Zhang, Xiangjing Hu, Jingyuan Zhang, Yating Zhang, Hui Wang, Lizhen Qu, and Zenglin Xu. 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)*, 2023.

[16] Tao Guo, Song Guo, and Junxiao Wang. 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, 2023. Association for Computing Machinery.

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

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

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

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

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

[22] Mitchell Wortsman, Suchin Gururangan, Shen Li, Ali Farhadi, Ludwig Schmidt, Michael Rabbat, and Ari S. Morcos. lo-fi: distributed fine-tuning without communication. *Transactions on Machine Learning Research (TMLR)*, (1), 2023.

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

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

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

[26] Jianyu Wang, Zachary Charles, Zheng Xu, Gauri Joshi, H Brendan McMahan, Maruan Al-Shedivat, Galen Andrew, Salman Avestimehr, Katharine Daly, Deepesh Data, et al. A field guide to federated optimization. *arXiv preprint arXiv:2107.06917*, 2021.

[27] Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. 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)*, 2018.

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

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