Adaptive Ranks for Personalized Federated Large Language Models under Parameter Budget Constraints

Jinhua Chen[1], Yuning Qiu[2], Franck Junior Aboya Messou[1],
Ruman Ahmed[3], Divya Jha[3], Koji Yoneda[3], Keping Yu[1]

[1]Hosei University, Japan
[2]RIKEN Center for Advanced Intelligence Project, Japan
[3]Sodick Co., Ltd., Japan

8th July

Contents

Motivation for Federated Large Language Models

- Large language models are now driving the interaction between artificial intelligence and humans across many sectors.
- In industry, the data used to train these models is private, so the raw data cannot be shared directly.
- Federated learning offers a solution, because it lets many parties train one shared model without ever moving their data.
- Different organizations also use language in different ways, so the model needs to be personalized.
- Finally, the very large scale of these models makes both communication and computation expensive.

The core requirement

We must balance privacy, efficiency, and adaptation on edge hardware that is highly heterogeneous.

LoRA is Efficient, but Lossy in Federation

LoRA (Low-Rank Adaptation) keeps the pre-trained weight frozen and only learns a small low-rank update, so the forward pass becomes:

$$h = W_0 x + \Delta W x = W_0 x + BAx$$

Here, W_0 is the frozen pre-trained weight, x and h are the input and output, $\Delta W = BA$ is the low-rank update, and B and A are the two trainable matrices.

- Because only B and A are trained, the number of trainable parameters and the communication cost both stay low.
- In a federated setting, the standard approach simply averages these updates across all clients.

The problem

Each client's update encodes its own data, so plain averaging causes the personalized information to be lost.

The Rank Dilemma

Edge devices differ greatly in bandwidth and computing power, but the rank is usually set to one uniform value for everyone.

- A small rank cannot express the needs of clients with strong personalization.
- A large rank exceeds the budget of the most resource-constrained devices.

Open gap

We need a mechanism that allocates ranks dynamically according to each client's personalization intensity, while staying within a fixed budget.

Related Work, Personalization

- Personalized federated learning and parameter selection methods, such as FedSelect [9] and FFT-MoE [10], mainly aim to improve local accuracy. So far, there is no unified framework that balances the strength of personalization against resource consumption under a strict budget.

Positioning

Our method fills this gap with a rank allocation that is dynamic, budget-constrained, and aware of personalization.

[9] R. Tamirisa, C. Xie, W. Bao, A. Zhou, R. Arel, and A. Shamsian, "FedSelect: Personalized federated learning with customized selection of parameters for fine-tuning," *CVPR*, pp. 23985–23994, 2024.
[10] G. Hu, Y. Teng, P. Wu, and N. Wang, "FFT-MoE: Efficient federated fine-tuning for foundation models via large-scale sparse MoE under heterogeneous edge," arXiv:2508.18663, 2025.

Federated Learning (FL)

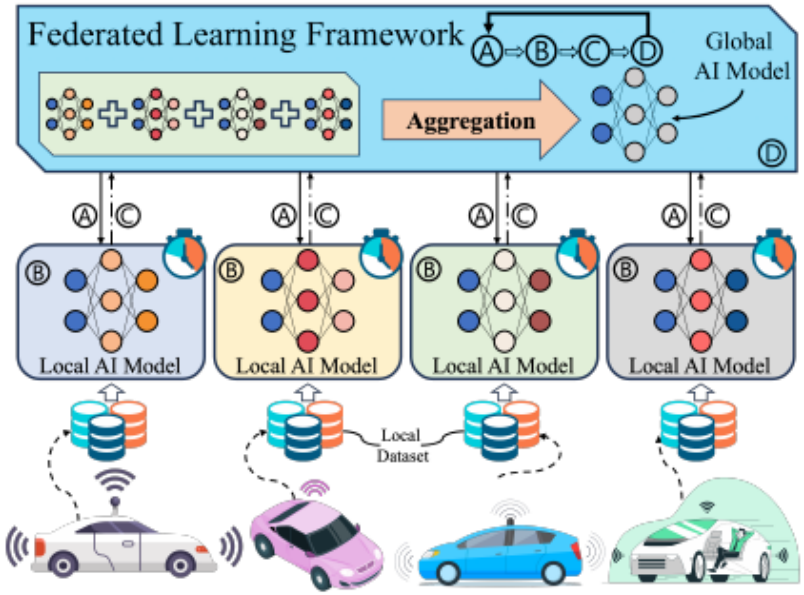


Figure 1: Federated Learning Framework

Federated Learning Steps:

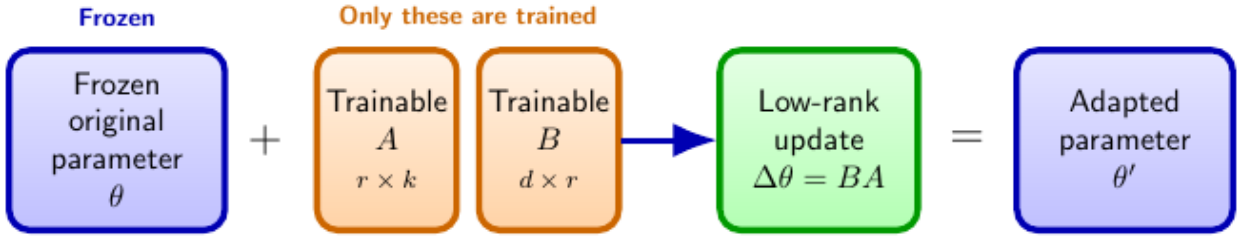
- Ⓐ A server initializes a global model and sends it to local clients.
- Ⓑ Local clients train local models with local datasets.
- Ⓒ Local clients upload the trained local models.
- Ⓓ The server aggregates all models and distributes the latest global model to the clients.

Benefit or Purpose: Ensures client privacy without exposing their data.

LoRA (Low-Rank Adaptation)

LoRA (Low-Rank Adaptation) is a parameter-efficient fine-tuning method for large language models.

- Full fine-tuning updates all LLM parameters, which is expensive for training, storage, and communication.
- LoRA freezes the original parameter matrix θ and only learns a small low-rank update $\Delta\theta$.
- **Benefit:** The model can adapt to a new task without updating the full LLM.



θ is frozen. Only A and B are trained. Therefore, LoRA reduces the number of trainable parameters.

In our FL setting, each client only fine-tunes and uploads lightweight LoRA updates $\Delta\theta_i^t$.

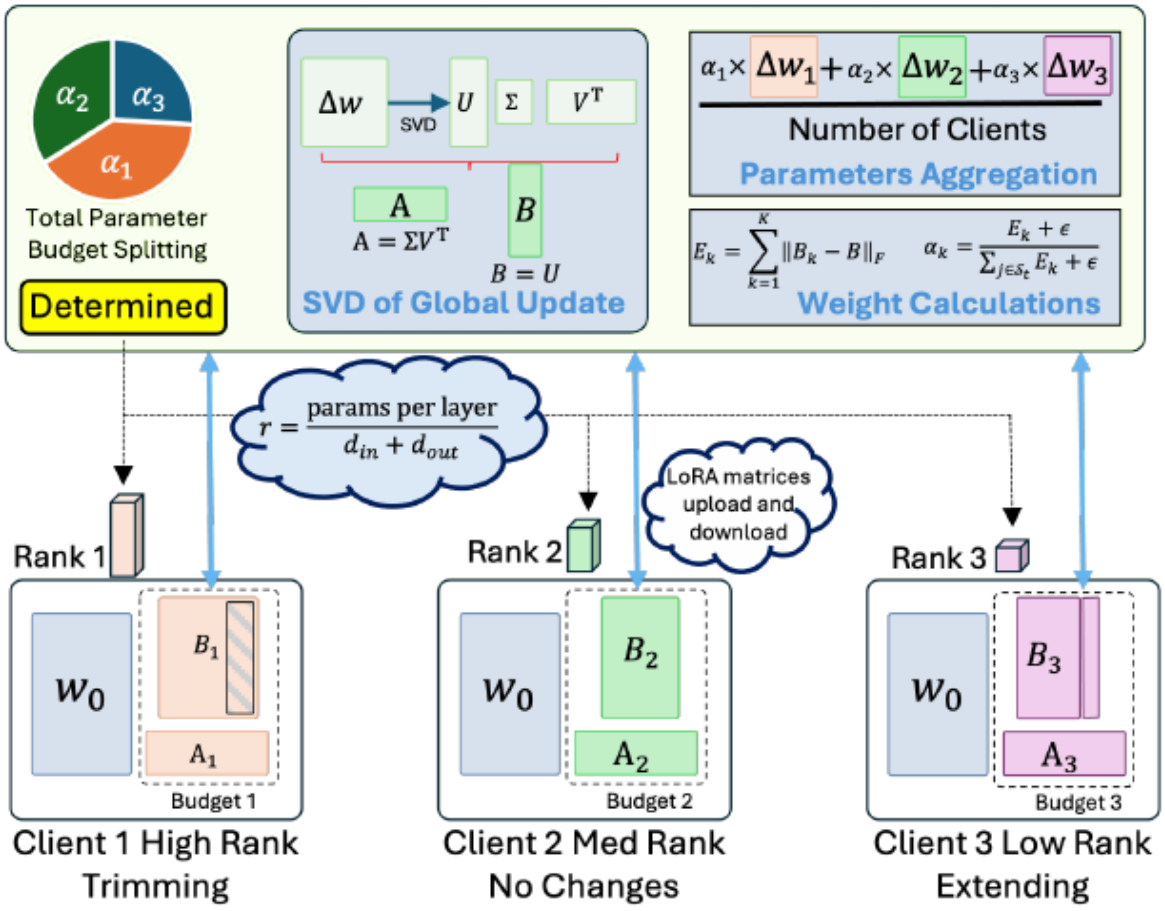
Adaptive Ranks Method Overview

In every round, the server runs three main operations.

1. It quantifies each client's personalization intensity using the Frobenius-norm divergence.
2. It fixes a total parameter budget and then assigns a rank to every client.
3. It aggregates the updates in the full weight space, and uses singular value decomposition to re-encode them back into each client's rank.

The overall effect is that the total cost stays fixed, while the budget flows to the clients and layers that need it most.

The Framework Pipeline



The server keeps the total parameter budget fixed, then redistributes rank capacity according to personalization strength.

The Framework Pipeline in One Round

1. The server measures each client's personalization energy from the divergence between local and global LoRA matrices.
2. The personalization energies become aggregation weights, so stronger client-specific adaptation has greater influence.
3. The fixed parameter budget is redistributed across clients, producing a tailored rank for each participant.
4. The server reconstructs the heterogeneous low-rank updates, averages them in the full weight space, and applies SVD to re-encode the next global and client-specific adapters.

The process moves capacity toward clients that need more adaptation while keeping the round-level communication cost unchanged.

The Algorithm in One Round

In each communication round the server runs four steps.

1. It broadcasts the current global matrices to the selected clients.
2. Each client trains locally on its own data, computes its personalization energy, and uploads both.
3. The server turns the energies into weights, sets the total budget, and derives a rank for every client.
4. For each layer, it aggregates the weighted updates, applies the decomposition, and re-encodes the global model and each client.

The budget stays fixed, and only the way it is distributed changes from one round to the next.

Experimental Setup, Model and Data

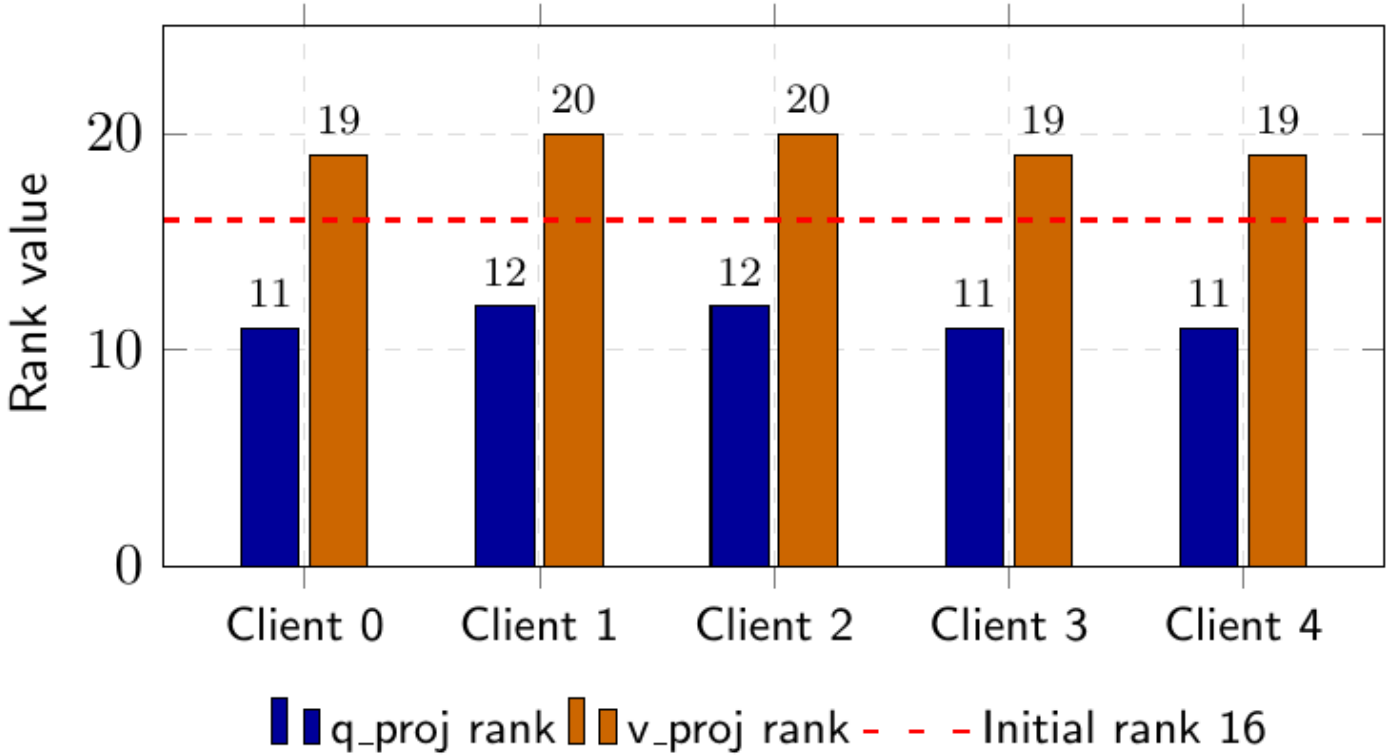
- The base model is Llama-3-8B-Instruct.
- The training data is the Alpaca dataset of fifty-two thousand instructions, split across five clients with a Dirichlet distribution at concentration zero point five, producing a non-independent and non-identically distributed partition.
- Evaluation uses the GLUE benchmark in a five-shot setting with the lm-evaluation-harness.

Experimental Setup, Training and Baselines

- LoRA adapters are placed on the query and value projections, with a reference rank of sixteen, a scaling factor of thirty-two, and per-client ranks between four and twenty-four.
- Training uses the AdamW optimizer with a learning rate of three times ten to the minus four, a batch size of sixteen, and fifty communication rounds.
- All experiments run on a single NVIDIA RTX 3090 graphics card.
- We compare against four baselines, namely FedAvgM, FedProx, FedAdagrad, and FedSA-LoRA, all sharing the same seed, data partition, and base model.

Result, Rank Allocation

Every layer begins at a uniform rank of sixteen, and the budget is then redistributed.



The value projection layers absorb more of the budget and grow to nineteen or twenty, while the query projection layers shrink to eleven or

Result, GLUE in the Five-shot Setting

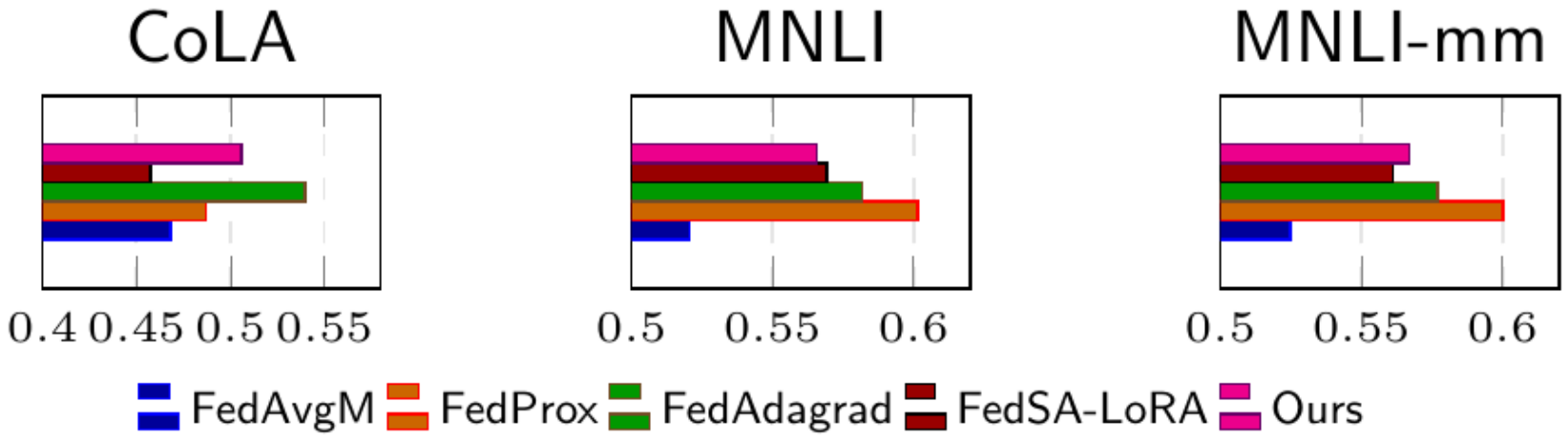
Table 1: Llama-3-8B-Instruct fine-tuned on Alpaca. A higher score is better, and a lower mean task ranking is better.

Method	CoLA	MNLI	MNLI-mm	MRPC	QNLI	QQP	RTE	SST-2	WNLI	Avg.	MTR
FedAvgM	0.4681	0.5206	0.5247	0.5980	0.5481	0.6888	0.6931	0.9312	0.5352	0.6120	4.67
FedProx	0.4868	**0.6016**	**0.6000**	0.4975	**0.6041**	0.7020	0.7437	0.8933	**0.6197**	0.6388	2.67
FedAdagrad	**0.5399**	0.5819	0.5770	0.6324	0.5907	0.7525	0.7437	0.9232	0.5493	0.6545	2.89
FedSA-LoRA	0.4573	0.5692	0.5612	**0.6691**	0.5832	0.7507	0.7473	**0.9346**	0.5775	0.6500	2.78
Ours	0.5057	0.5658	0.5669	0.6520	0.5927	**0.7576**	**0.7509**	0.9266	0.5775	**0.6551**	**2.22**

- Our method reaches the best average score of zero point six five five one, and the best mean task ranking of two point two two, among all methods.
- It performs best on QQP and RTE, and it recovers strongly on CoLA and MRPC.

GLUE Detail: CoLA, MNLI, MNLI-mm

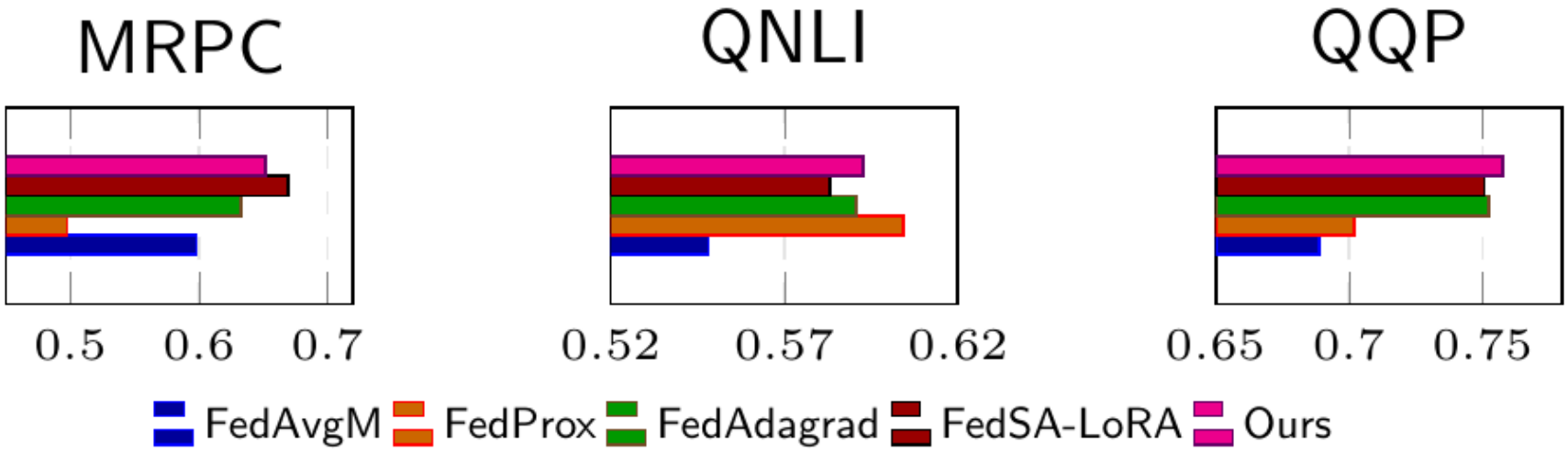
CoLA (Corpus of Linguistic Acceptability) measures whether a sentence is grammatically acceptable. **MNLI** and **MNLI-mm** (Multi-Genre Natural Language Inference, matched and mismatched) test whether a hypothesis follows from a premise across diverse text genres.



Our method ranks second on CoLA and stays competitive on both MNLI tasks, showing stable language understanding across genres.

GLUE Detail: MRPC, QNLI, QQP

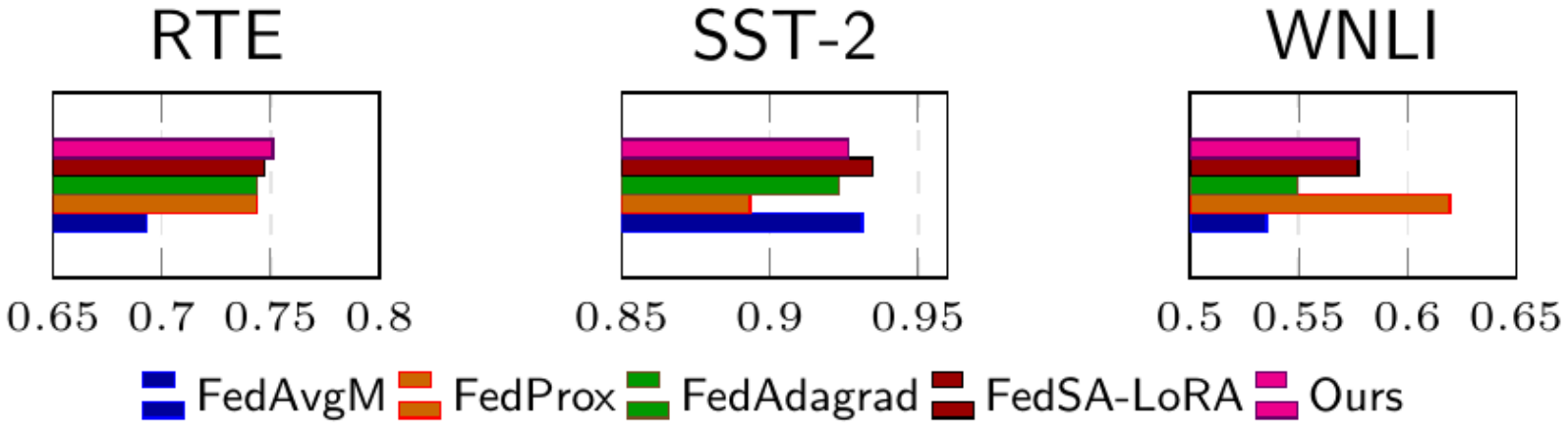
MRPC (Microsoft Research Paraphrase Corpus) tests whether two sentences are paraphrases of each other. **QNLI** (Question NLI) asks whether a sentence answers a given question. **QQP** (Quora Question Pairs) detects duplicate questions.



Our method achieves the best score on QQP and ranks second on MRPC and QNLI, showing strength in semantic similarity tasks.

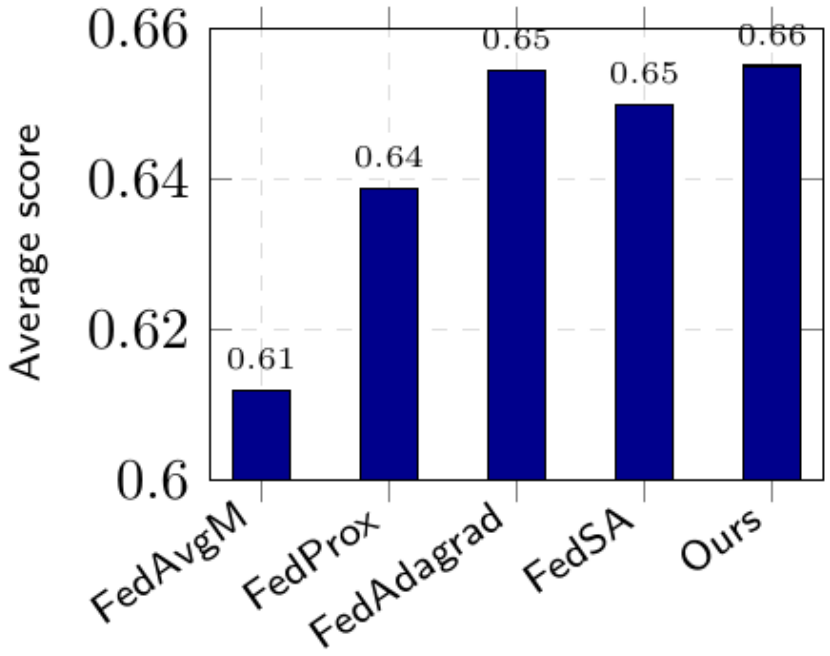
GLUE Detail: RTE, SST-2, WNLI

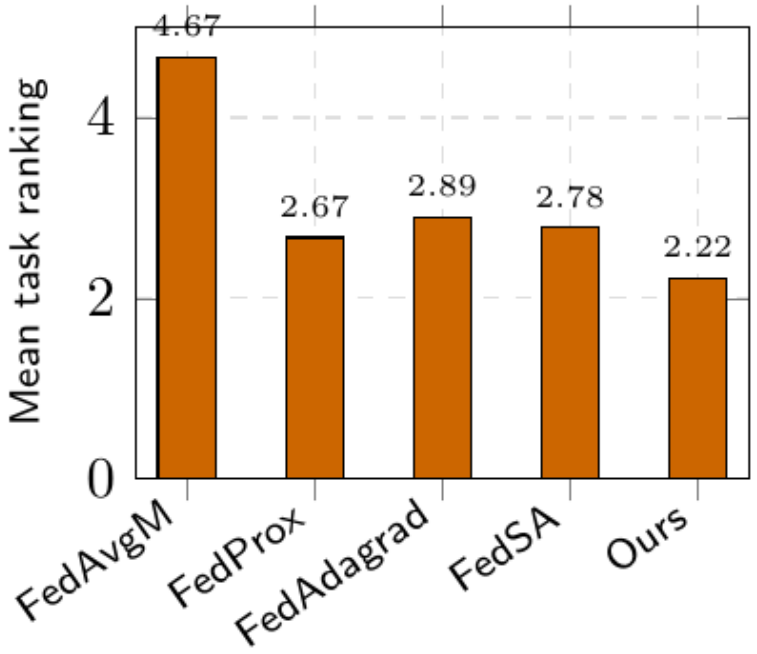
RTE (Recognizing Textual Entailment) determines whether one sentence entails another. **SST-2** (Stanford Sentiment Treebank) classifies movie review sentences as positive or negative. **WNLI** (Winograd NLI) tests pronoun resolution and commonsense reasoning.



Our method achieves the best score on RTE, and remains competitive on SST-2 and WNLI.

Result, Average Score and Stability





Because the budget is redistributed adaptively, the model does not over-fit any single task distribution. It stays within the top three on most tasks, and it achieves the lowest, and therefore the best, mean task ranking.

Analysis

- By measuring each update's divergence and moving the budget toward layers with strong personalization, the method avoids the information loss of uniform averaging.
- The baselines tend to spike on one task and then drop on another. For example, FedProx scores high on MNLI but only zero point four nine seven five on MRPC, whereas our method stays in the top three across the board.
- On SST-2 our score of zero point nine two six six is just below the best of zero point nine three four six, a gap within zero point zero zero eight. Sentiment classification is fairly uniform across clients, so personalization offers only a small gain here.

Overall, adaptive ranks preserve global generalization while also keeping each client's personalization, all under a fixed budget.

Conclusion

- We presented an adaptive aggregation framework for federated fine-tuning of large language models under personalization and resource heterogeneity.
- The framework uses the Frobenius-norm divergence to quantify personalization, and from this it allocates the rank budget dynamically.
- It aggregates heterogeneous low-rank updates in the full weight space through singular value decomposition.
- On the GLUE benchmark, it achieves the best average score of zero point six five five one, and the best mean task ranking of two point two two.

The broader impact

Matching each client's capacity to its real need lowers the entry barrier for participants with limited compute, and reduces the communication and energy cost of each round.

Discussion

Thank you for your time and attention

Questions and answers are welcome