

A Technical Appendices and Supplementary Material

A.1 Related Works

Federated Learning. Federated learning (FL) enables a distributed set of clients to collaboratively train a single global model using their local data [25]. The primary challenges in FL are data heterogeneity and computational heterogeneity induced by the statistical and hardware differences across clients, respectively. Several works aim to address these challenges by altering either the client training procedure or the server aggregation algorithm. FedProx [22], SCAFFOLD [18], and FedDyn [1] add a corrective regularization term to each client’s local objective—whether a proximal penalty (FedProx), control-variate correction (SCAFFOLD), or dynamic regularizer (FedDyn)—to keep local updates from drifting too far from the global model, a phenomena more often referred to as “client drift”. An alternative approach is to adjust the server aggregation scheme to reduce client drift. For example, FedVARP [47] incorporates historical client updates into the current round’s aggregation step to reduce the variance caused by partial client participation and heterogeneity. FedExp [48] varies the server learning rate by using an extrapolation rule that increases the server step size when consecutive aggregated updates point in similar directions and shrinks it when they diverge.

Parameter-Efficient Fine-Tuning. Recently, many works have adopted the *pretrain-then-fine-tune* framework, in which a general-purpose large language model (LLM) is adapted to a smaller downstream task [10, 29, 44, 46]. The excessive computational cost of fine-tuning LLMs has led to parameter-efficient fine-tuning (PEFT) methods that fine-tune a fraction of the overall parameters of the model. Adapter tuning [45], BitFit [41], and low-rank adaptation (LoRA) [15] have emerged as effective PEFT methods that can significantly reduce the number of parameters while maintaining task performance. Since its conception, many works have improved upon the initial LoRA formulation in various ways. Works such as QLoRA [43] and LoftQ [49] quantize the pretrained model weights to further improve the memory efficiency of LoRA-based fine-tuning. LoRA-XS [5], LoRA-SB [50], and MoRA [17] introduce an additional LoRA parameter while freezing *both* the model backbone and the original LoRA parameters during fine-tuning. HydraLoRA [33], LoRAMoE [11], and MoLE [51] employ a mixture-of-experts architecture to traditional LoRA-based fine-tuning frameworks.

PEFT Methods for FL. As newer applications look to use on-device data to fine-tune LLMs, PEFT methods for FL have become increasingly relevant. FedPETuning [52], FedPrompt [53], and FedIT [39] incorporate adapter tuning, prompt tuning, and LoRA into federated frameworks, respectively. More recently, methods like FFA-LoRA [32], FedEx-LoRA [30], Fed-SB [31], and RoLoRA [42] optimize LoRA in homogeneous-compute FL by addressing the inexactness in LoRA aggregation caused by separately averaging the **B** and **A** parameters. An orthogonal direction is explored by works such as HetLoRA [8], FlexLoRA [4], and FLoRA [36] that address computational heterogeneity in LoRA-based FL fine-tuning by allowing clients to train LoRA parameters with different ranks. However, an examination of cross-device fine-tuning remains limited in this context as methods like FLoRA have communication costs that scale linearly with the number of clients and communication rounds, making fine-tuning particularly difficult in large-scale FL systems. The goal of RAVAN is to design a PEFT method for FL that addresses data and computational heterogeneity, while scaling effectively to cross-device settings with a large number of clients and communication rounds. In this way, we overcome shortcomings in prior works and enable resource-aware fine-tuning.

A.2 Broader Impacts

RAVAN enables accurate, efficient fine-tuning in a federated setting. While RAVAN has not yet been integrated into a real-world FL system, we identify two potential impacts of utilizing RAVAN:

- **Edge Data for LLM Fine-Tuning:** RAVAN provides an opportunity for LLMs to utilize the data collected by edge devices. As edge applications become increasingly critical, capitalizing on these specialized datasets can make these applications more effective without forfeiting data locality.
- **Efficiency and Improved Performance:** LLM fine-tuning is an expensive, energy-consuming procedure. RAVAN improves the efficiency of the process while retaining performance. Applied to realistic FL training regimes, RAVAN addresses some of these prior concerns.

RAVAN should be implemented with safeguards to prevent misuse, privacy leaks, and harmful content. While RAVAN does not exacerbate these issues, they remain concerns in LLM usage more broadly.

A.3 Experimental Settings

Hyperparameters and Optimization Details. In this section, we highlight hyperparameter choices used in our experiments that were not discussed in Section 4. These descriptions, in addition to the provided code, should aid in the reproducibility of the stated results.

Table 7: FL hyperparameter settings used for each model–dataset pair.

	Model/Dataset				
	ViT-B-16/CIFAR-100	ViT-B-16/SVHN	T5-Base/20 Newsgroups	T5-Base/MRQA	LLaMA3.2-1B/GLUE
Batch Size	32	32	32	32	16
Max Sequence Length	–	–	256	256	128
Local Iterations	50	50	50	50	50
Communication Rounds	50	50	100	20	20
Total Epochs (per round)	1	1	1	1	1

For RAVAN and each baseline, we run a learning rate hyperparameter sweep across the values $\{5e-5, 1e-5, 5e-4, 1e-4, 5e-3, 1e-3, 5e-2, 1e-2, 5e-2\}$ and choose the most performant learning to represent in our results. Table 8 represents the optimal choices for each baseline in all settings. The following results each use the ADAM optimizer with momentum set to 0.9.

Table 8: Optimal learning rate configurations for all baselines.

(a) Lower parameter budget / I.I.D. clients.

Method	Model/Dataset				
	ViT-B-16/CIFAR-100	ViT-B-16/SVHN	T5-Base/20 Newsgroups	T5-Base/MRQA	LLaMA3.2-1B/GLUE
FedIT	5×10^{-3}	1×10^{-3}	1×10^{-3}	1×10^{-3}	1×10^{-4}
FedEx-LoRA	1×10^{-3}	1×10^{-3}	1×10^{-3}	1×10^{-3}	1×10^{-4}
FFA-LoRA	1×10^{-2}	1×10^{-2}	1×10^{-2}	5×10^{-3}	1×10^{-3}
Fed-SB	1×10^{-3}	5×10^{-3}	5×10^{-3}	1×10^{-3}	1×10^{-4}
RAVAN	5×10^{-4}	5×10^{-4}	5×10^{-4}	1×10^{-4}	5×10^{-5}

(b) Higher parameter budget / I.I.D. clients.

Method	Model/Dataset				
	ViT-B-16/CIFAR-100	ViT-B-16/SVHN	T5-Base/20 Newsgroups	T5-Base/MRQA	LLaMA3.2-1B/GLUE
FedIT	5×10^{-3}	1×10^{-3}	1×10^{-3}	5×10^{-4}	1×10^{-4}
FedEx-LoRA	1×10^{-3}	1×10^{-3}	1×10^{-3}	5×10^{-4}	1×10^{-4}
FFA-LoRA	1×10^{-2}	1×10^{-2}	1×10^{-2}	1×10^{-2}	1×10^{-3}
Fed-SB	1×10^{-3}	1×10^{-3}	5×10^{-3}	5×10^{-4}	1×10^{-4}
RAVAN	5×10^{-4}	5×10^{-4}	1×10^{-4}	1×10^{-4}	5×10^{-5}

(c) Lower parameter budget / non-I.I.D. clients.

Method	Model/Dataset			
	ViT-B-16/CIFAR-100	ViT-B-16/SVHN	T5-Base/20 Newsgroups	T5-Base/MRQA
FedIT	5×10^{-3}	1×10^{-3}	1×10^{-3}	1×10^{-3}
FedEx-LoRA	1×10^{-3}	1×10^{-3}	1×10^{-3}	1×10^{-3}
FFA-LoRA	1×10^{-2}	1×10^{-2}	1×10^{-2}	5×10^{-3}
Fed-SB	5×10^{-4}	5×10^{-3}	1×10^{-3}	5×10^{-4}
RAVAN	5×10^{-4}	5×10^{-4}	5×10^{-4}	1×10^{-4}

(d) Higher parameter budget / non-I.I.D. clients.

Method	Model/Dataset			
	ViT-B-16/CIFAR-100	ViT-B-16/SVHN	T5-Base/20 Newsgroups	T5-Base/MRQA
FedIT	5×10^{-3}	1×10^{-3}	1×10^{-3}	5×10^{-4}
FedEx-LoRA	1×10^{-3}	5×10^{-4}	1×10^{-3}	5×10^{-4}
FFA-LoRA	1×10^{-2}	1×10^{-2}	1×10^{-2}	5×10^{-3}
Fed-SB	5×10^{-4}	1×10^{-3}	1×10^{-3}	5×10^{-4}
RAVAN	5×10^{-4}	5×10^{-4}	1×10^{-4}	1×10^{-4}

529 **Baseline Descriptions.** We provide details for each of the baselines used in our experiments. We
 530 highlight how each baseline initializes, trains, and communicates the individual LoRA parameters.

- 531 • **FedIT:** FedIT initializes the LoRA parameters with $\mathbf{B}^{(0)} = \mathbf{0}$ and $\mathbf{A}^{(0)} \sim \mathcal{N}(0, \sigma^2)$. In communi-
 532 cation round t , each client $c \in \mathcal{C}^{(t)}$ locally trains the LoRA parameters resulting in local parameters
 533 $\mathbf{B}_c^{(t)}, \mathbf{A}_c^{(t)}$. After communicating these parameters back to the central server, the server performs
 534 the following aggregation to generate the new global model:

$$\mathbf{B}^{(t+1)} = \frac{1}{|\mathcal{C}^{(t)}|} \sum_{c \in \mathcal{C}^{(t)}} \mathbf{B}_c^{(t)}, \quad \mathbf{A}^{(t+1)} = \frac{1}{|\mathcal{C}^{(t)}|} \sum_{c \in \mathcal{C}^{(t)}} \mathbf{A}_c^{(t)} \quad (9)$$

- 535 • **FedEx-LoRA:** FedEx-LoRA initializes the LoRA parameters with $\mathbf{B}^{(0)} = \mathbf{0}$ and $\mathbf{A}^{(0)} \sim \mathcal{N}(0, \sigma^2)$.
 536 In communication round t , each client $c \in \mathcal{C}^{(t)}$ locally trains the LoRA parameters resulting in
 537 local parameters $\mathbf{B}_c^{(t)}, \mathbf{A}_c^{(t)}$. To address the exact aggregation issue, the server updates both the
 538 global LoRA parameters as well as the model backbone:

$$\begin{aligned} \mathbf{B}^{(t+1)} &= \frac{1}{|\mathcal{C}^{(t)}|} \sum_{c \in \mathcal{C}^{(t)}} \mathbf{B}_c^{(t)}, \quad \mathbf{A}^{(t+1)} = \frac{1}{|\mathcal{C}^{(t)}|} \sum_{c \in \mathcal{C}^{(t)}} \mathbf{A}_c^{(t)} \\ \mathbf{W}^{(t+1)} &= \mathbf{W}^{(t)} + \left(\frac{1}{|\mathcal{C}^{(t)}|} \sum_{c \in \mathcal{C}^{(t)}} \mathbf{B}_c^{(t)} \mathbf{A}_c^{(t)} - \frac{1}{|\mathcal{C}^{(t)}|} \sum_{c \in \mathcal{C}^{(t)}} \mathbf{B}_c^{(t)} \cdot \frac{1}{|\mathcal{C}^{(t)}|} \sum_{c \in \mathcal{C}^{(t)}} \mathbf{A}_c^{(t)} \right) \end{aligned} \quad (10)$$

539 While this ensures exact updates in every round, the updated model backbone $\mathbf{W}^{(t+1)}$ also has to
 540 be communicated from the central server, increasing the communication overhead of the procedure.

- 541 • **FFA-LoRA:** FFA-LoRA initializes the LoRA parameters with $\mathbf{B}^{(0)} = \mathbf{0}$ and $\mathbf{A} \sim \mathcal{N}(0, \sigma^2)$.
 542 However, the \mathbf{A} parameter remains frozen at initialization and is never locally trained by the clients
 543 and communicated throughout the procedure. Thus, the only update throughout training is the
 544 update to the LoRA \mathbf{B} parameter:

$$\mathbf{B}^{(t+1)} = \frac{1}{|\mathcal{C}^{(t)}|} \sum_{c \in \mathcal{C}^{(t)}} \mathbf{B}_c^{(t)} \quad (11)$$

- 545 • **Fed-SB:** Fed-SB uses three LoRA parameters which, for the sake of consistency with prior notation,
 546 we call $\mathbf{B} \in \mathbb{R}^{d \times r}$, $\mathbf{H} \in \mathbb{R}^{r \times r}$, $\mathbf{A} \in \mathbb{R}^{r \times d}$. The weight update is reparameterized as $\mathbf{B}\mathbf{H}\mathbf{A}$. To
 547 initialize the LoRA parameters, Fed-SB performs an initial round of full-parameter fine-tuning to
 548 obtain a full-parameter weight update $\Delta \mathbf{W}_{\text{full}}$. The weight update is decomposed using SVD to
 549 get $\Delta \mathbf{W}_{\text{full}} = \mathbf{U}\Sigma\mathbf{V}^\top$. \mathbf{B} , \mathbf{H} , and \mathbf{A} are then initialized as $\mathbf{B} = \mathbf{U}[:, 1:r]$, $\mathbf{H}^{(0)} = \Sigma[1:r, 1:r]$,
 550 $\mathbf{A} = \mathbf{V}^\top[1:r, :]$. \mathbf{B} and \mathbf{A} are frozen at initialization, so the only update is the following:

$$\mathbf{H}^{(t+1)} = \frac{1}{|\mathcal{C}^{(t)}|} \sum_{c \in \mathcal{C}^{(t)}} \mathbf{H}_c^{(t)} \quad (12)$$

- 551 • **FlexLoRA:** FlexLoRA allows each client $c \in \mathcal{C}^{(t)}$ to train LoRA parameters with client-specific
 552 ranks r_c . To aggregate the LoRA parameters, the server performs SVD on $\frac{1}{|\mathcal{C}^{(t)}|} \sum_{c \in \mathcal{C}^{(t)}} \mathbf{B}_c^{(t)} \mathbf{A}_c^{(t)} =$
 553 $\mathbf{U}\Sigma\mathbf{V}^\top$. To redistribute the LoRA parameters back to the clients, the central server sends each
 554 client $c \in \mathcal{C}^{(t+1)}$ the following LoRA parameters:

$$\mathbf{B}_c^{(t+1)} = \mathbf{U}[:, 1:r_c] \Sigma[1:r_c, 1:r_c], \quad \mathbf{A}_c^{(t+1)} = \mathbf{V}^\top[1:r_c, :] \quad (13)$$

- 555 • **HetLoRA:** Let r_{\max} be the highest rank supported by any client. Each client pads its local
 556 parameters to this common shape (with zeros in the unused columns and rows) before upload, so
 557 aggregation is still dimensionally consistent. The server weights the individual LoRA parameters
 558 based on their relative Frobenius norms:

$$\begin{aligned} S_c^{(t)} &= \|\mathbf{B}_c^{(t)} \mathbf{A}_c^{(t)}\|_F, \quad p_c^{(t)} = \frac{S_c^{(t)}}{\sum_{c \in \mathcal{C}^{(t)}} S_c^{(t)}} \\ \mathbf{B}^{(t+1)} &= \sum_{c \in \mathcal{C}^{(t)}} p_c^{(t)} \mathbf{B}_c^{(t)}, \quad \mathbf{A}^{(t+1)} = \sum_{c \in \mathcal{C}^{(t)}} p_c^{(t)} \mathbf{A}_c^{(t)}, \end{aligned} \quad (14)$$

559 The server then truncates the new global LoRA parameters $\mathbf{B}^{(t+1)}$ and $\mathbf{A}^{(t+1)}$ for each client
 560 $c \in \mathcal{C}^{(t+1)}$ so that $\mathbf{B}_c^{(t+1)} = \mathbf{B}^{(t+1)}[:, 1:r_c]$ and $\mathbf{A}_c^{(t+1)} = \mathbf{A}^{(t+1)}[1:r_c, :]$.

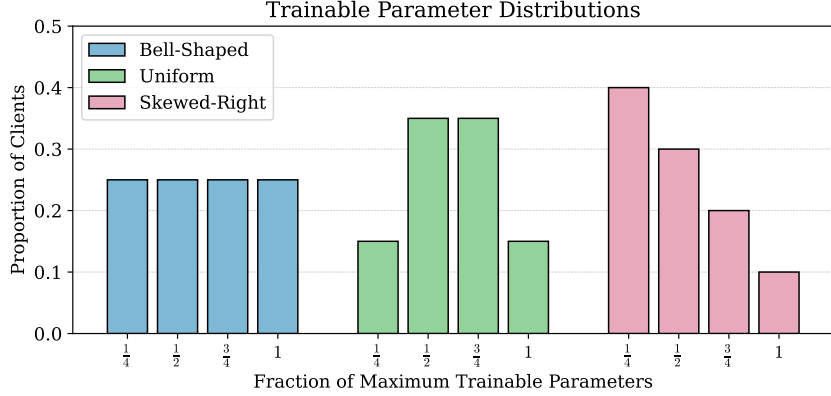


Figure 5: Fraction of clients assigned to each trainable parameter budget in each distribution. Skewed-left is omitted because it is never used in our experiments.

Computational Heterogeneity Setup. To emulate computational heterogeneity in our FL setup, we vary the number of trainable parameters at each client. Let N_{\max} denote the largest number of trainable parameters that any client can afford. Each client c is constrained to a trainable parameter budget $N_c \in \{\frac{1}{4}, \frac{1}{2}, \frac{3}{4}, 1\}N_{\max}$. This value is held constant throughout the FL procedure to mirror fixed hardware limits. N_c simply determines the trainable parameter budget per-client; all other hyperparameters are identical to the compute-homogeneous experiments. The bar plot in Figure 5 provides a visual summary of the client mix. Uniform serves as a neutral baseline where there is an equal proportion of clients at every trainable parameter budget; bell-shaped concentrates clients around medium ranks, reflecting the case where most devices have moderate capability; skewed-right stresses the system by placing a large share of the population at the lowest rank, leaving only a small fraction of high-capacity contributors. HetLoRA, FlexLoRA, and RAVAN each accommodate clients with different trainable-parameter budgets in distinct ways:

- **HetLoRA and FlexLoRA:** The LoRA rank for each client c is $r_c = \lfloor (N_c / N_{\max}) \cdot r_{\max} \rfloor$ where r_{\max} is the maximum rank trained by any client. Since the number of trainable parameters scales linearly with the rank, this scaling ensures that every client keeps its update within the allotted budget N_c while allowing higher-capacity devices to contribute proportionally higher-rank updates.
- **RAVAN:** RAVAN uses H LoRA heads per weight matrix. A client with budget N_c , fine-tunes only $\lfloor (N_c / N_{\max}) \cdot H \rfloor$ heads and leaves the remaining heads frozen (e.g. for $N_c = \frac{1}{4}N_{\max}$ the client trains one quarter of the heads).

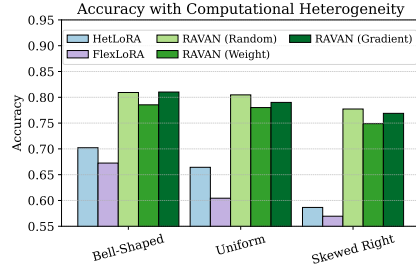
Table 9: Layers equipped with LoRA adapters in each model backbone.

Model	LoRA Target Modules
ViT-B-16	query, value
T5-Base	SelfAttention.q, SelfAttention.v
LLaMA3.2-1B	q_proj, v_proj

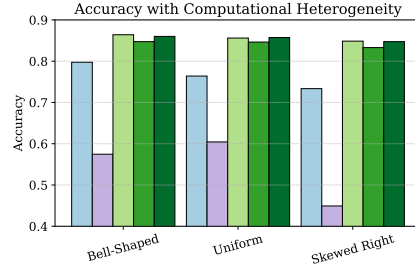
LoRA Implementation Details. For every model backbone, we insert LoRA adapters only in the self-attention projection matrices. The exact parameters for which we apply LoRA are described in Table 9. All other parameters are frozen and do not have associated LoRA parameters.

Compute Details and Cluster Description. All experiments were executed on a GPU cluster managed by SLURM. Each training job used a single NVIDIA V100 32GB GPU with 256 GB RAM. Our environment used Pytorch 2.5.1 and Huggingface 4.47.1 for all experiments. With this setup, each experimental run took ≈ 1 GPU hour with ViT-B-16 for both CIFAR-100 and SVHN, ≈ 2 GPU hours with T5-Base for 20 Newsgroups, ≈ 3 GPU hours with T5-Base for MRQA, and ≈ 2 GPU hours with LLaMA3.2-1B for each GLUE subtask. All baselines were trained with identical hardware, batch sizes, optimizers, and communication rounds to ensure fair comparison.

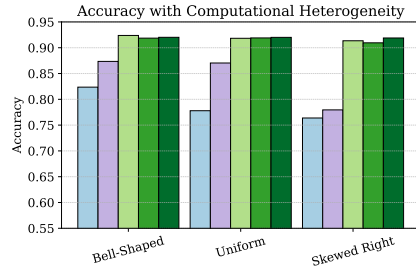
590 A.4 Additional Experiments



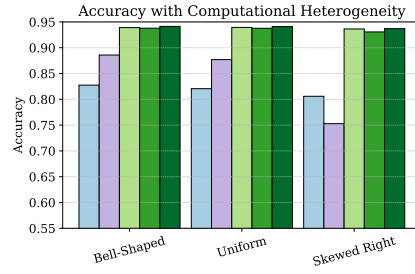
(a) CIFAR-100 / lower parameter budget



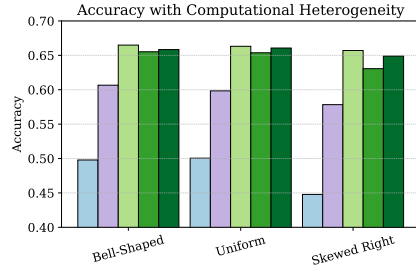
(b) CIFAR-100 / higher parameter budget



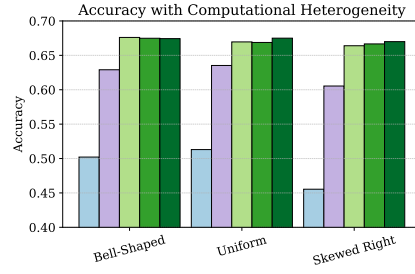
(c) SVHN / lower parameter budget



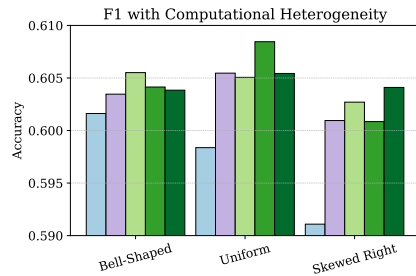
(d) SVHN / higher parameter budget



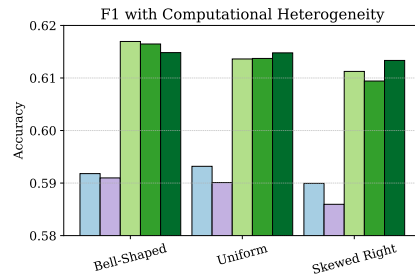
(e) 20 Newsgroups / lower parameter budget



(f) 20 Newsgroups / higher parameter budget



(g) MRQA / lower parameter budget



(h) MRQA / higher parameter budget

Figure 6: Impact of computational heterogeneity on baselines and RAVAN across four datasets. Each row shows a single dataset (left: lower parameter budget, right: higher parameter budget). All settings match the descriptions from Section 4.

591 **Computational Heterogeneity Experiments.** We evaluate all methods with 20 non-I.I.D. clients
 592 whose usable parameter budgets are drawn from three distributions. Across vision and language tasks,
 593 RAVAN’s variants consistently outperform the competing LoRA baselines. These results underscore
 594 RAVAN’s robustness to computational heterogeneity across different tasks and parameter budgets.

Table 10: LoRA ranks under lower vs. higher parameter budgets.

Method	Lower Budget	Higher Budget
FedIT	32	64
FedEx-LoRA	32	64
FFA-LoRA	64	128
Fed-SB	221	313
RAVAN	110	156

Table 11: Accuracy comparison on GLUE benchmark with LLaMA3.2-1B.

(a) 20 clients / lower parameter budget

Method	MNLI-MM	MNLI-M	QNLI	QQP	SST-2	RTE	Average
FedIT	84.24	84.62	82.74	85.96	94.61	65.70	82.97
FedEx-LoRA	84.15	84.70	82.74	86.07	94.61	65.34	82.94
FFA-LoRA	85.05	85.78	82.07	84.40	94.38	62.46	82.36
Fed-SB	84.88	85.23	82.84	84.23	94.95	67.15	83.21
RAVAN	85.24	85.65	84.00	86.11	95.18	67.15	83.90

(b) 20 clients / higher parameter budget

Method	MNLI-MM	MNLI-M	QNLI	QQP	SST-2	RTE	Average
FedIT	83.74	83.24	87.72	85.60	95.30	68.95	84.09
FedEx-LoRA	83.95	83.41	87.79	85.65	95.41	70.04	84.38
FFA-LoRA	85.27	84.69	89.51	87.10	95.18	68.23	85.00
Fed-SB	85.85	84.76	89.53	86.09	94.95	66.79	84.66
RAVAN	86.20	85.34	90.35	87.22	95.18	70.04	85.72

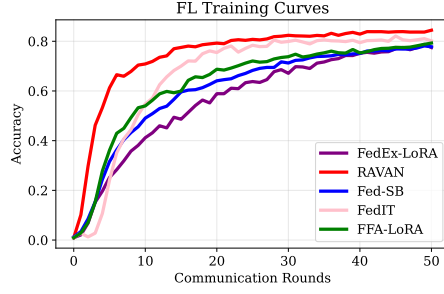
(c) 50 clients / lower parameter budget

Method	MNLI-MM	MNLI-M	QNLI	QQP	SST-2	RTE	Average
FedIT	84.22	84.24	87.53	85.87	94.61	61.73	83.03
FedEx-LoRA	84.25	84.15	87.77	85.81	94.61	62.09	83.11
FFA-LoRA	85.92	85.05	89.33	87.40	95.30	60.29	83.88
Fed-SB	85.71	84.65	88.05	86.08	94.15	64.98	83.94
RAVAN	86.03	85.53	88.91	86.95	95.41	62.09	84.15

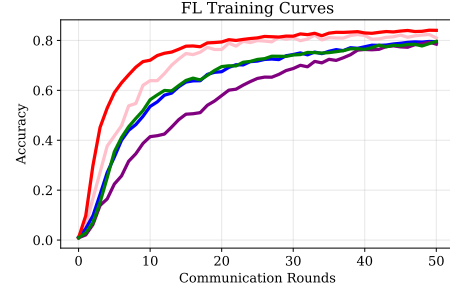
(d) 50 clients / higher parameter budget

Method	MNLI-MM	MNLI-M	QNLI	QQP	SST-2	RTE	Average
FedIT	84.66	84.26	88.38	85.87	95.18	63.58	83.66
FedEx-LoRA	84.74	84.02	88.50	85.82	95.41	58.12	82.77
FFA-LoRA	85.35	84.64	87.21	87.20	94.50	61.37	83.38
Fed-SB	85.91	85.24	87.68	86.30	93.58	67.15	84.31
RAVAN	86.17	85.35	88.87	87.39	95.87	64.62	84.71

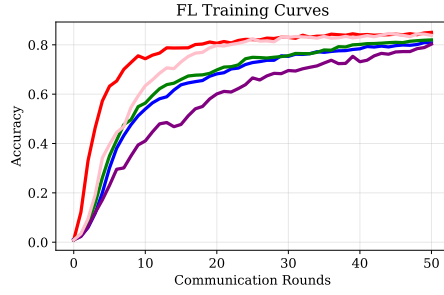
LLaMA Experiments. In these experiments, we use the same hyperparameter settings described in Section 4 but vary the number of total clients and the the ranks of each baseline. Across all four GLUE configurations, RAVAN consistently matches or exceeds the performance of the strongest PEFT baselines using LLaMA3.2-1B (see Table 11). Additionally, while other PEFT baselines vary in performance across settings, RAVAN’s performance remains consistent in all configurations. This suggests that RAVAN maintains robust performance, demonstrating its ability to scale effectively to larger models and diverse FL scenarios.



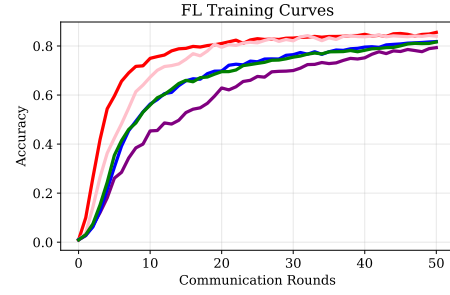
(a) CIFAR-100 / 20 clients / lower parameter budget



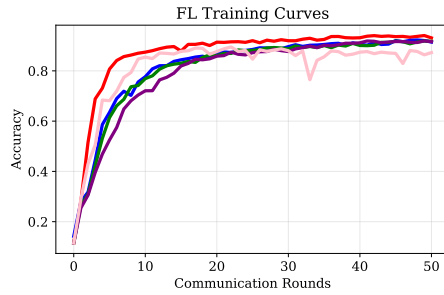
(b) CIFAR-100 / 50 clients / lower parameter budget



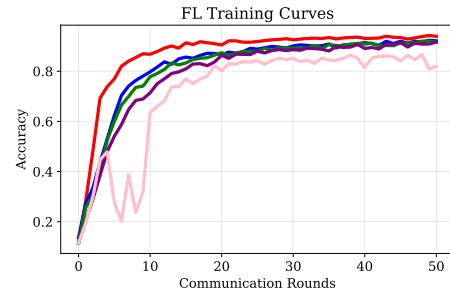
(c) CIFAR-100 / 20 clients / higher parameter budget



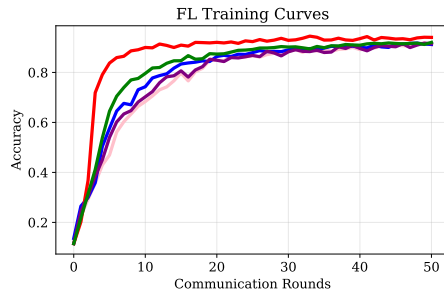
(d) CIFAR-100 / 50 clients / higher parameter budget



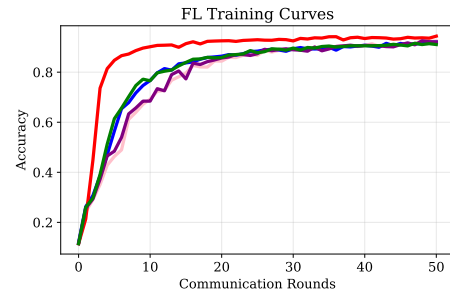
(e) SVHN / 20 clients / lower parameter budget



(f) SVHN / 50 clients / lower parameter budget



(g) SVHN / 20 clients / higher parameter budget



(h) SVHN / 50 clients / higher parameter budget

Figure 7: FL training curves for CIFAR-100 and SVHN for all benchmarks.

Training Curves. Figure 7 displays the training curves for the various PEFT benchmarks on CIFAR100 and SVHN using a varying number of I.I.D. clients and trainable parameter budgets. In comparison to the other PEFT methods, RAVAN converges faster and to a better overall performance, suggesting that it requires fewer communication rounds to reach optimal performance.

Appendix References

- [41] Elad Ben Zaken, Yoav Goldberg, and Shauli Ravfogel. BitFit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio, editors, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1–9, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-short.1. URL <https://aclanthology.org/2022.acl-short.1/>.
- [42] Shuangyi Chen, Yue Ju, Hardik Dalal, Zhongwen Zhu, and Ashish J Khisti. Robust federated finetuning of foundation models via alternating minimization of loRA. In *Workshop on Efficient Systems for Foundation Models II @ ICML2024*, 2024. URL <https://openreview.net/forum?id=xT0acYbg0F>.
- [43] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. QLoRA: Efficient finetuning of quantized LLMs. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL <https://openreview.net/forum?id=OUIFPHEgJU>.
- [44] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Tamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL <https://aclanthology.org/N19-1423/>.
- [45] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for NLP. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2790–2799. PMLR, 09–15 Jun 2019. URL <https://proceedings.mlr.press/v97/houlsby19a.html>.
- [46] Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. In Iryna Gurevych and Yusuke Miyao, editors, *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1031. URL <https://aclanthology.org/P18-1031/>.
- [47] Divyansh Jhunjhunwala, Pranay Sharma, Aushim Nagarkatti, and Gauri Joshi. Fedvarp: Tackling the variance due to partial client participation in federated learning. In *Uncertainty in Artificial Intelligence*, pages 906–916. PMLR, 2022.
- [48] Divyansh Jhunjhunwala, Shiqiang Wang, and Gauri Joshi. Fedexp: Speeding up federated averaging via extrapolation. In *The Eleventh International Conference on Learning Representations*, 2023. URL <https://openreview.net/forum?id=IPrzNbddXV>.
- [49] Yixiao Li, Yifan Yu, Chen Liang, Nikos Karampatziakis, Pengcheng He, Weizhu Chen, and Tuo Zhao. Loftq: LoRA-fine-tuning-aware quantization for large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL <https://openreview.net/forum?id=LzPWWPAdY4>.
- [50] Kaustubh Ponkshe, Raghav Singhal, Eduard Gorbunov, Alexey Tumanov, Samuel Horvath, and Praneeth Vepakomma. Initialization using update approximation is a silver bullet for extremely efficient low-rank fine-tuning, 2025. URL <https://arxiv.org/abs/2411.19557>.
- [51] Xun Wu, Shaohan Huang, and Furu Wei. Mixture of loRA experts. In *The Twelfth International Conference on Learning Representations*, 2024. URL <https://openreview.net/forum?id=uWvKBCYh4S>.

- 654 [52] Zhuo Zhang, Yuanhang Yang, Yong Dai, Qifan Wang, Yue Yu, Lizhen Qu, and Zenglin Xu.
655 FedPETuning: When federated learning meets the parameter-efficient tuning methods of pre-
656 trained language models. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors,
657 *Findings of the Association for Computational Linguistics: ACL 2023*, pages 9963–9977,
658 Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.
659 findings-acl.632. URL <https://aclanthology.org/2023.findings-acl.632/>.
- 660 [53] Haodong Zhao, Wei Du, Fangqi Li, Peixuan Li, and Gongshen Liu. Fedprompt: Communication-
661 efficient and privacy-preserving prompt tuning in federated learning. In *ICASSP 2023 - 2023*
662 *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages
663 1–5, 2023. doi: 10.1109/ICASSP49357.2023.10095356.