LoRA in Federated Learning

Amirparsa Safari, Mohammadreza Javadi

Previous papers

- FL-TAC
- FedLEASE
- FedEx-LoRA
- LoRA-FAIR
- FRLoRA
- FLASC
- FedSA-LoRA

This presentation

- FedLoRU
- Flex-LoRA
- CE-LoRA
- FSLoRA
- FLoRA

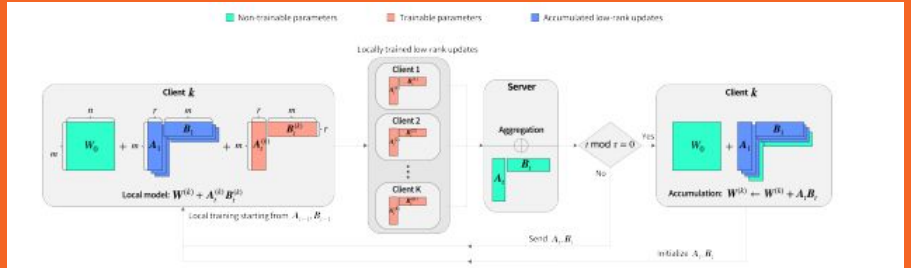
FedLoRU

Algorithm 1 FedLoRU. $\boldsymbol{W}$ is a model, $\boldsymbol{A}_0, \boldsymbol{B}_0$ are initial low-rank update matrices, α is a scaling factor, τ is an accumulation cycle, T is the total training round.

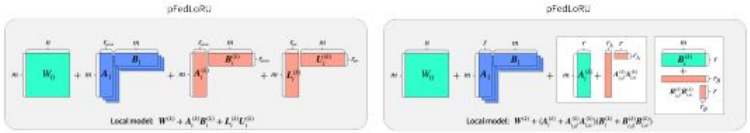
Require: $\boldsymbol{W}$, $\boldsymbol{A}_0, \boldsymbol{B}_0$, α, τ, T.
  Initialize: Server sends $\boldsymbol{W}$ to each client.
  for $t = 1, \cdots, T$ **do**
    Server selects M clients $\mathcal{K}_M$ and distributes $\boldsymbol{A}_{t-1}, \boldsymbol{B}_{t-1}$ to clients in $\mathcal{K}_M$.
    for each client $k \in \mathcal{K}_M$ **do**
      Local training: Find $\boldsymbol{A}_t^{(k)}, \boldsymbol{B}_t^{(k)}$ by solving (6) starting from $\boldsymbol{A}_{t-1}, \boldsymbol{B}_{t-1}$.
      Send $\boldsymbol{A}_t^{(k)}, \boldsymbol{B}_t^{(k)}$ to the server.
    end for
    Server aggregation: $\boldsymbol{A}_t \leftarrow \sum_{k \in \mathcal{K}_M} p^{(k)} \boldsymbol{A}_t^{(k)}$, $\boldsymbol{B}_t \leftarrow \sum_{k \in \mathcal{K}_M} p^{(k)} \boldsymbol{B}_t^{(k)}$.
    if $t \bmod \tau = 0$ **then**
      Server distributes $\boldsymbol{A}_t, \boldsymbol{B}_t$ to all clients .
      Each client k updates its local copy of the global model: $\boldsymbol{W} \leftarrow \boldsymbol{W} + \alpha \boldsymbol{A}_t \boldsymbol{B}_t$.
    end if
  end for
  Return: $\boldsymbol{W} + \alpha \sum_{t=1:\, t \bmod \tau = 0}^{T} \boldsymbol{A}_t \boldsymbol{B}_t$.

$$\boldsymbol{A}_t^{(k)},\ \boldsymbol{B}_t^{(k)} = \underset{\boldsymbol{A},\, \boldsymbol{B}}{\arg\min}\, f^{(k)}(\boldsymbol{W} + \alpha \boldsymbol{A}\boldsymbol{B}) \tag{6}$$

where α is a fixed scaling hyperparameter. At the end of each iteration, the server collects $\boldsymbol{A}_t^{(k)}$ and $\boldsymbol{B}_t^{(k)}$ and aggregates them by averaging: $\boldsymbol{A}_t = \sum_{k \in \mathcal{K}_M} p^{(k)} \boldsymbol{A}_t^{(k)}$, $\boldsymbol{B}_t = \sum_{k \in \mathcal{K}_M} p^{(k)} \boldsymbol{B}_t^{(k)}$. After the aggregation, the server broadcasts $\boldsymbol{A}_t$ and $\boldsymbol{B}_t$ to the clients, who continue local training using these matrices.



(a) Flow-chart of FedLoRU algorithm



(b) Low-rank factorization methods in pFedLoRU and mFedLoRU

- how federated learning can be made more communication-efficient and scalable while handling client heterogeneity, without significantly sacrificing model accuracy
- Algorithm Design: FedLoRU employs iterative low-rank factorization (matrices A and B) for client updates, with server-side accumulation of these low-rank matrices to build a high-rank model
- Experiments: Conducted on datasets like Fashion-MNIST, CIFAR-10/100, and language models (LLaMA2-3B), comparing against baseline FL algorithms (FedAvg, FedLoRA, FedHM).

What things got better?

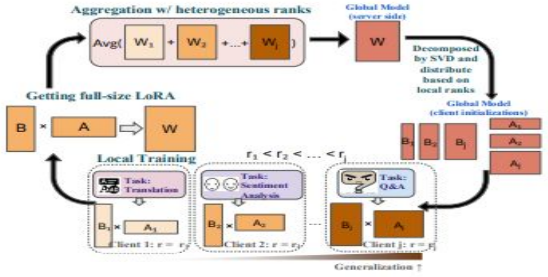
- Communication efficiency improved by reducing parameter transmission from quadratic to linear in model size
- Scalability improved as the number of clients increased
- Robustness to heterogeneity (data and model) improved via FedLoRU variants
- Performance in large-client (cross-device) settings improved compared to baseline FL algorithms
- Test loss and accuracy matched or exceeded baselines in several datasets and tasks, including LLM fine-tuning

What got worse?

- In some scenarios with small client populations, performance can be slightly lower than full-rank methods
- Computational burden may increase due to low-rank factorization at each client
- Theoretical guarantees rely on assumptions that may not fully reflect real-world data
- Effectiveness across all forms of extreme non-iid data is not fully validated

Flex-LoRA

What was the study trying to answer?
It aimed to address how federated fine-tuning of large language models can effectively handle heterogeneous client resources and data distributions. It specifically sought a way to avoid restricting all clients to the lowest resource level while preserving privacy and scalability.
What did it answer?
It showed that allowing clients to use different local LoRA ranks and aggregating them via FlexLoRA improves global model generalization. It demonstrated that SVD-based redistribution can successfully combine heterogeneous updates without harming performance.



3 Methodology of FlexLoRA

3.1 Intrinsic Dimension and Generalization

Fine-tuning LLMs to enhance task-specific performance inevitably encounters cost of reduced generalization ability: a trade-off supported by the "no-free-lunch" theorem and empirical evidence usually called "alignment tax" of LLM [40, 31]. The generalization capability of LLMs is influenced by complexity of applied tasks and their solution spaces, which can be characterized by the concept of an intrinsic dimension – typically far smaller than the total number of model parameters [1].

The insight of intrinsic dimension informs the design of LoRA to fine-tune LLMs' pre-trained weights in a parameter-efficient manner, utilizing compact and low-rank matrices. Specifically, matrices $A \in \mathbb{R}^{r\times p}, B \in \mathbb{R}^{d\times r}$ are introduced, where r denotes the rank that encapsulates intrinsic dimension. These matrices form a low-rank approximation for tuning original weights W_0 as $h = W_0x + sBAx$, where x is the input of the parameter to be tuned, h is the output, and s is a scaling constant. Previous studies show that different ranks produce weights with attributes particularly tailored to specific downstream tasks [16, 18]. Consequently, the rank value plays a critical role in not only task-specific solution subspaces but also in determining a model's ability to generalize to various tasks.

In scenarios where clients have highly heterogeneous task and resource distributions, a uniform LoRA rank usually does not suffice for model performance, especially in its zero-shot generalization ability for unseen clients and tasks. Employing a small LoRA rank potentially leads to under-fitting in a global context by capturing only a subset of task-specific features, while a large rank is usually infeasible due to the "bucket effect" of existing FL solutions constrained by least-resourced clients.

FlexLoRA emerges as a solution to this dilemma by dynamically adjusting the rank in response to the variability in local client resources. By increasing the LoRA rank for clients with greater resources to contribute more global knowledge, FlexLoRA enhances the model's ability to generalize across diverse data distributions without sacrificing local performance accuracy. This strategy allows for federated fine-tuning of LLMs to navigate between the extremes of task-specific optimization and generalization to unseen clients and tasks.

3.2 Aggregation with Heterogeneous Ranks

Traditional FL methods like FedAvg aggregate local LoRA weights by computing a weighted average of the decomposed matrices A and B as $B_g = (\sum_{i=1}^{m} n^i B_l^i)/(\sum_{i=1}^{m} n^i)$, $A_g = (\sum_{i=1}^{m} n^i A_l^i)/(\sum_{i=1}^{m} n^i)$, where B_g, A_g are the global LoRA decomposed matrices, and B_l^i, A_l^i are the local LoRA decomposed matrices of i-th client, n^i is the size of the i-th client's local training dataset, m is the number of FL clients. However, this scheme is restricted by the lowest LoRA rank among participating clients for aggregation compatibility, which makes it hard to capture the full diversity of client contributions and fully utilize ample client resources.

FlexLoRA takes a different yet simple approach to enable decomposed matrix with different LoRA ranks to be mixed together. Specifically, it first forms a low-rank approximation of the LoRA matrix for each client, W_l^i, before computing the weighted average: $W_g = (\sum n^i W_l^i)/(\sum_{i=1}^{m} n^i) = (\sum n^i sB_l^i A_l^i)/(\sum_{i=1}^{m} n^i)$.

After the weighted average with heterogeneous LoRA ranks, the resulting global LoRA weight W_g is decomposed using SVD. Then the SVD components U, Σ, V are redistributed to clients in a low-rank approximation that preserves as much information of W_g as possible meanwhile based on clients' local resources characterized by r^i:

$$\text{SVD}(W_g) = U\Sigma V^T, \qquad W_g^i = U[:, : r^i]\Sigma[: r^i, : r^i]V[: r^i, :]^T \approx W_g,$$

where U, Σ, and V^T are the SVD components of W_g, the r^i within $[]$ indicates the indexing operator of each client to select their singular vectors corresponding to top r^i singular values. As a result, client i receives the aggregated knowledge W_g^i from server and incorporates W_g^i into its local LoRA weight $W_g^i = sB_g^i A_g^i$ with $B_g^i = U[:, : r^i]\Sigma[: r^i, : r^i]/s$, and $A_g^i = V[: r^i, :]^T$.

The local training then proceeds as similar to those in standard FL approaches, using W_l^i as the local weights to be tuned. This aforementioned process is repeated until convergence is achieved or a predetermined number of rounds is completed.

What things got better?

- Global model generalization across diverse NLP tasks improved
- Zero-shot performance on unseen clients increased (up to ~4% improvement)
- Resource utilization improved by allowing capable clients to contribute more via higher LoRA ranks
- Convergence speed improved when FlexLoRA was integrated into existing FL methods
- Robustness improved across different resource distributions, especially in heavy-tail scenarios
- Communication and computation efficiency remained scalable with negligible SVD overhead

What got worse?

- The approach has not yet been fully validated in real-world large-scale deployments
- Effectiveness depends on the availability of sufficient computational resources for high-rank clients
- Theoretical guarantees rely on assumptions (e.g., Lipschitz continuity) that may not hold across all models or data distributions
- Performance may be affected by biases in simulated task or resource distributions

CE-LoRA

What was the study trying to answer?

- fine-tune large pre-trained foundation models in federated learning settings while reducing communication costs.
- performance under non-IID and heterogeneous client data distributions.
- personalized federated fine-tuning, while preserving privacy
- make federated adaptation of large models scalable and practical for distributed clients with limited data.

What did it answer?

- CE-LoRA, a federated fine-tuning framework combining tri-matrix LoRA factorization with personalized aggregation.
- showed how personalized aggregation based on dataset and model similarity can address data heterogeneity.
- showed transmitting only low-rank matrices significantly reduces communication overhead.
- showed that CE-LoRA improves accuracy, convergence speed, and privacy compared to existing federated LoRA-based methods.
- empirically validated robustness under increasing client numbers and higher heterogeneity levels.

Algorithm 1 Training process of CE-LoRA

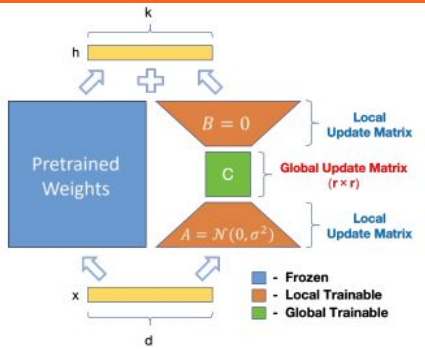
Inputs: Client set $\{P_i\}_{i=1}^m$; Communication round T; The pre-trained model parameters W; Tri-LoRA parameters A_i, B_i, C_i for $L_i, i = 1, ..., m$; The local dataset D_i on client P_i.

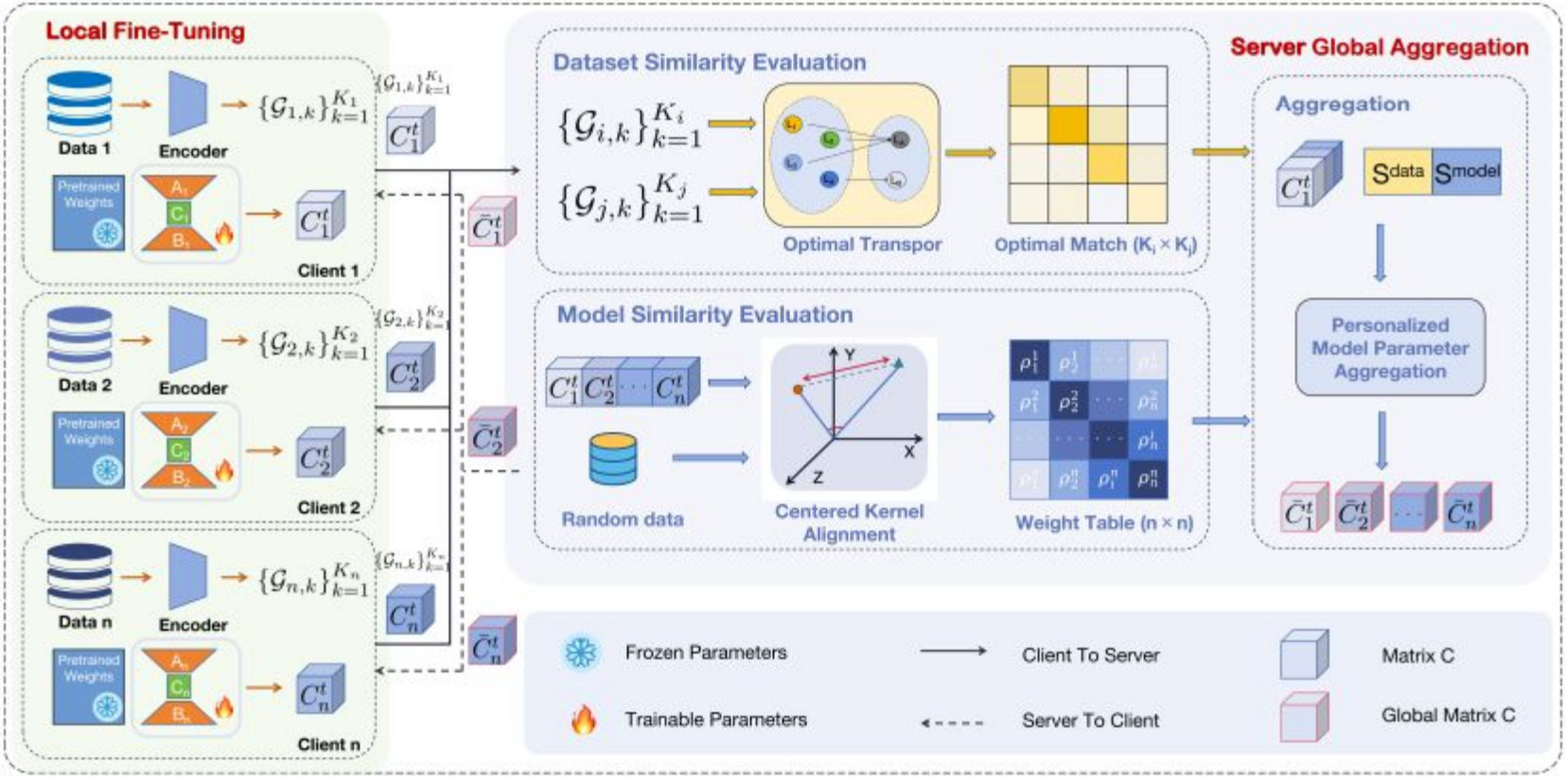
for $t = 1$ to T **do**
  for each *client* $P_i \in \{P_j\}_{j=1}^m$ in parallel **do**
    Fine-tune local LoRA to learn A_i^t, C_i^t, B_i^t via

$$\min_{A_i, C_i, B_i} f_i(A_i, C_i, B_i | W, \bar{C}_i^{t-1}, D_i);$$

    Send C_i^t to the server;
    Keep A_i^t and B_i^t locally for next round use.
  end for
  Server receives $\{C_i^t\}_{i=1}^m$ from the clients;
  Server conducts personalized model parameter aggregation to calculate $\{\bar{C}_i^t\}_{i=1}^m$ for the clients (refer to §III-C);
  Server send $\{\bar{C}_i^t\}_{i=1}^m$ to the corresponding clients.
end for

Outputs: $\{A_i, C_i, B_i\}_{i=1}^m$.




Local Fine-Tuning
Data 1
Encoder
Pretrained Weights
Client 1
Data 2
Encoder
Pretrained Weights
Client 2
Data n
Encoder
Pretrained Weights
Client n
Dataset Similarity Evaluation
Optimal Transpor
Optimal Match (Ki × Kj)
Model Similarity Evaluation
Random data
Centered Kernel Alignment
Weight Table (n × n)
Server Global Aggregation
Aggregation
Sdata
Smodel
Personalized Model Parameter Aggregation
Frozen Parameters
Trainable Parameters
Client To Server
Server To Client
Matrix C
Global Matrix C

What things got better?

- Communication efficiency improved dramatically, with up to 1024× reduction in transmitted parameters.
- Model accuracy and generalization improved across NLP and computer vision tasks.
- Convergence speed increased compared to baseline federated fine-tuning methods.
- Robustness to non-IID data distributions improved through personalized aggregation.
- Privacy protection improved by mitigating gradient-based data reconstruction attacks.
- Performance remained stable as the number of clients increased and heterogeneity intensified.
-

What got worse?

- Computational overhead increased due to similarity calculations that scale quadratically with the number of clients.
- The method depends on accurate similarity metrics, and errors in these metrics can reduce personalization quality.
- Applicability is limited by evaluation on a fixed set of datasets and models.
- Task and model heterogeneity are not directly addressed in the current approach.

FSLoRA

What was the study trying to answer?

- Fine-tune large language models directly on resource-constrained devices within a FL setting.
- handle heterogeneity in device computation, memory, and communication capabilities during federated fine-tuning.
- How to reduce communication and computation costs while maintaining model accuracy and convergence.
- How to design a theoretically grounded, resource-adaptive federated LoRA-based fine-tuning method

What did it answer?

- It proposed FSLoRA, a federated fine-tuning algorithm that integrates a sketching mechanism into LoRA modules.
- It showed how selectively updating submatrices of LoRA parameters enables adaptation to diverse device resources.
- It provided theoretical convergence analysis linking sketching ratios to convergence behavior.
- It demonstrated through experiments that FSLoRA improves accuracy and efficiency compared to baseline federated LoRA methods.
- It showed that devices with limited resources can still benefit from larger global model ranks via sketching.

Algorithm 1 Federated Sketching LoRA (FSLoRA)

Require: Base model $\mathbf{W}_0$, LoRA modules $\mathbf{B}_0$ and $\mathbf{A}_0$, learning rate γ, and sketching set $\{\mathcal{S}_i\}_{i=1}^N$

for $t = 0, 1, \ldots, T-1$ **do**
  Server samples sketching matrices $\{\mathbf{S}_i^t \sim \mathcal{S}_i\}_{i=1}^N$ and sends them back to the devices
  Server broadcasts the current global LoRA modules to the devices
  for $h = 0, 1, \ldots, H-1$ **do**
    Devices update the local LoRA modules via (4)
  end for
  Devices upload the non-zero columns of $(\mathbf{B}_i^{t,H} - \mathbf{B}_i^{t,0})$ and the non-zero rows $(\mathbf{A}_i^{t,H} - \mathbf{A}_i^{t,0})$ to the server
  Server updates the global LoRA modules via (5)
end for

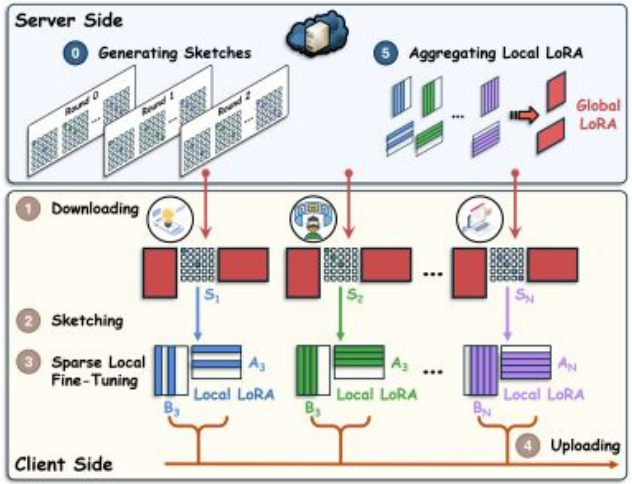


Figure 1: An illustration of our proposed methodology where the server maintains a pair of global LoRA modules while the devices adaptively update submatrices of the global LoRA modules through sketching during each round.

What things got better?

- Model accuracy improved across multiple datasets and models compared to baseline methods.
- Resource efficiency improved by reducing computation and communication on constrained devices.
- Convergence speed improved with appropriate sketching ratios.
- Adaptability to heterogeneous device capabilities increased.
- Stability of training was maintained across different global rank and sketching configurations.
- Privacy-preserving, on-device fine-tuning became more feasible.
-

What got worse?

- The method has not been validated on real-world hardware deployments.
- Theoretical guarantees rely on assumptions that may not fully reflect real-world conditions.
- Performance–efficiency trade-offs depend on manually chosen sketching ratios.
- The impact of extreme heterogeneity in devices and data is not fully explored.
-

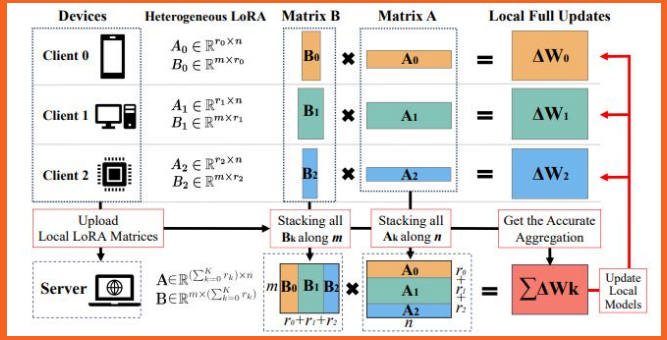
FLoRA

What was the study trying to answer?

- What limitations exist in current federated LoRA-based fine-tuning methods, especially FedIT.
- How to perform accurate aggregation of LoRA adapters in federated learning without introducing noise.
- How to support heterogeneous client resources and different LoRA ranks during federated fine-tuning.
- How to enable efficient, privacy-preserving fine-tuning of large language models in distributed settings.

What did it answer?

- Identified that independently averaging LoRA adapters (A and B) in FedIT is mathematically incorrect and introduces aggregation noise.
- proposed FLORA, a stacking-based aggregation method that ensures mathematically correct, noise-free aggregation.
- showed that stacking supports heterogeneous LoRA ranks across clients without requiring rank alignment or padding.
- demonstrated through experiments that FLORA outperforms other baselines across models, datasets, and benchmarks.
- provided theoretical support for the convergence and correctness of the stacking-based aggregation.



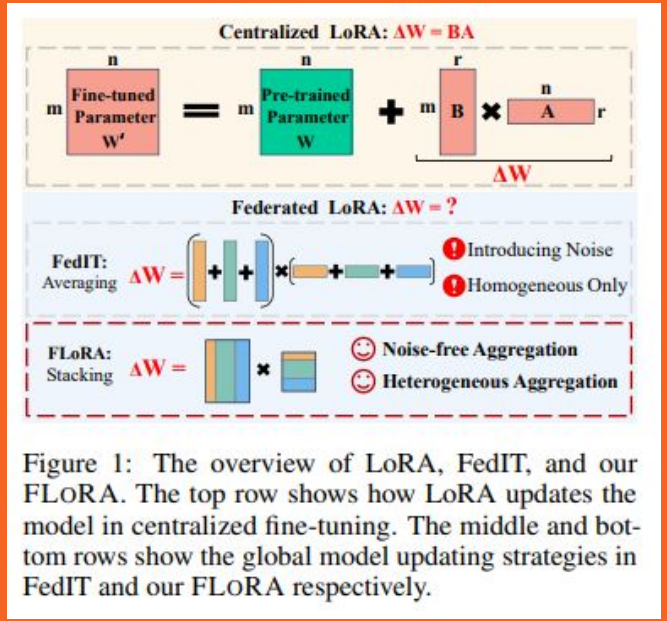


Figure 1: The overview of LoRA, FedIT, and our FLoRA. The top row shows how LoRA updates the model in centralized fine-tuning. The middle and bottom rows show the global model updating strategies in FedIT and our FLoRA respectively.

What things got better?

- Aggregation accuracy improved by eliminating noise from independent averaging.
- Model performance improved, often exceeding FedIT and sometimes even centralized fine-tuning.
- Support for heterogeneous client resources and LoRA ranks improved.
- Convergence speed and stability improved across homogeneous and heterogeneous settings.
- Robustness to data and system heterogeneity increased.
- Efficient participation of diverse clients was enabled without increasing full-model communication.

What got worse?

- Communication overhead increased slightly due to stacking larger LoRA modules.
- Potential privacy risks increased due to transmitting stacked modules.
- Dependence on scaling factor choices introduced sensitivity in fine-tuning performance.
- Theoretical guarantees rely on assumptions that may not fully hold in practice.