Proceedings of the 4th International Conference on Machine Learning and Automation
DOI: 10.54254/2755-2721/2026.38681

Toward Federated Fine-Tuning of Large Language Models: Progress, Challenges, and Future Directions

Zilang He

School of Computing and Data Science, The University of Hong Kong, Hong Kong, China
hezilang667733@gmail.com

Abstract. Large language models (LLMs) have been extensively used in downstream tasks, and the successful implementation of such tasks typically involves fine-tuning on domain-specific or user-specific data. As these data are generally not centralized but rather distributed among clients, federated learning (FL) of LLMs, also known as FedLLM, has emerged as a promising paradigm to decentralize LLM adaptation. Nevertheless, it is still challenging to directly apply traditional FL to LLM fine-tuning due to the amplification of model scale, client heterogeneity, communication cost, and privacy risks. This survey offers an overview of FedLLM fine-tuning from methodological and evaluation aspects. This paper first presents the fundamentals of FL, LLM fine-tuning, and their combination. Then this paper categorizes representative works into three groups: efficiency, heterogeneity and personalization, and privacy and security. In addition, this paper summarizes commonly used datasets and evaluation benchmarks to show how existing methods are evaluated under different client partitions and task settings. Finally, according to the given review, this paper identifies future directions such as resource-adaptive collaboration, split federated fine-tuning, continual personalization, and privacy-preserving safety alignment. All things considered, this survey takes FedLLM fine-tuning as its main point of attention, which gives a well-structured reference for researchers and practitioners to get to know about this field and find out potential research paths.

Keywords: Federated LLM Fine-Tuning, Parameter-Efficient Fine-Tuning, Heterogeneous Aggregation, Personalization

1. Introduction

Large Language Models (LLMs) are now being used in various application fields due to their high capabilities in language understanding, multimodal perception, text generation and logical reasoning [1]. The practical deployment of these models in a variety of downstream situations requires, however, domain-specific or user-specific fine-tuning on large amounts of local data. Such information is usually distributed among users, devices or institutions that cannot be centralized directly because of the privacy risks, issues with the ownership of the data, and the limitations of regulations. Federated learning (FL) for LLMs (FedLLM) is an attractive framework to solve this problem by allowing clients to adjust the model locally and share only the updates of the model with the central server [2]. In such a manner, FedLLM makes it possible for LLMs to learn using the

© 2026 The Authors. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (https://creativecommons.org/licenses/by/4.0/).

DOI: 10.54254/2755-2721/2026.38681

decentralized and heterogeneous sources of data and at the same time minimize the risk of disclosure of raw data.

Nevertheless, it is not easy to implement FL in LLM fine-tuning. The scale of the parameters and the architecture of the LLMs are huge and therefore require a lot of computing resources, memory, and communication, which can be quite burdensome for clients with limited resources that are being trained on these models [3]. Moreover, distributed data tend to be very diverse, thus model aggregation and personalization will be more challenging as compared to traditional centralized fine-tuning [4]. These issues have led to the emergence of studies on FedLLM fine-tuning where full-parameter fine-tuning, parameter-efficient fine-tuning (PEFT), instruction fine-tuning, personalized adaptation, and alignment-based fine-tuning have been proposed [5].

FedLLM fine-tuning has been a topic of interest in the recent years, and some studies have been done on various technical issues concerning decentralization of LLM adaptation. The initial research works like FedIT [6] discussed federated instruction tuning by training the LLMs on distributed data of instructions and retaining raw data locally whereas Ye et al. proposed OpenFedLLM [7], which is a general framework to train LLMs using federation and later developed FedLLM-Bench [8] to facilitate standardized testing under realistic federation conditions. In order to make it more efficient, there are parameter-efficient approaches like LoRA [9] and its derivatives that can reduce the number of trainable parameters through low-rank learning and FedKSeed [10] that can minimize the communication cost of full-parameter tuning of federated systems by sharing random seeds. To adapt to heterogeneous and personalized settings, FLoRA [11] is designed as an approach to aggregate heterogeneous LoRA with stacked client-side low-rank updates, and FedALT [4] is designed to allow for individual and shared LoRA modules, so that local knowledge can be transferred to other clients. Privacy and security have also been discussed, with PPFedIT [12] protecting local instruction data by generating synthetic data and training in parameter-isolated mode. In general, these studies indicate that FedLLM fine-tuning has become a research topic of great importance and is slowly establishing a systematic research framework.

Despite the fact that a number of recent surveys have studied FedLLM fine-tuning and similar topics, most of them do not offer a systematic and focused analysis. Wu et al. [2] and Yao et al. [5] are concerned with the general research advancement and broad issues whereas Woisetschläger et al. [3] and Yang et al. [13] are more interested in specific challenges or paradigms. On the contrary, this survey assumes FedLLM fine-tuning as the main theme and seeks to offer a systematic guide to researchers and practitioners to get a better insight into this area and determine the future directions of research.

The rest of this survey is arranged in the following way. Section 2 presents the initial ideas about federated learning, LLM fine-tuning, and FedLLM fine-tuning. In section 3, representative studies are reviewed and the main methodological developments in this field are summarized. Section 4 talks about datasets and evaluation benchmarks of FedLLM fine-tuning. Section 5 discusses major challenges and future research prospects. Lastly, Section 6 concludes this survey.

2. Preliminary

This section presents some basic notions of federated learning, LLM fine-tuning, and FedLLM fine-tuning as a basis for the following review.

DOI: 10.54254/2755-2721/2026.38681

2.1. Federated learning

Federated learning (FL) [14] is a common distributed learning paradigm, where multiple clients can jointly train a shared model under a parameter-server architecture with the raw data being kept local. Federated Averaging (FedAvg) [15] is a classic FL algorithm, in which the server performs weighted aggregation using the uploaded updates of selected clients and then broadcasts the aggregated parameters back to them. Let us assume that $\mathscr{S}_t$ represent the set of clients that participate in round t, and each client k has nk samples. The server aggregation may be expressed as:

$$\theta^{t+1} = \sum_{k\in\mathscr{S}_t} \frac{n_k}{\sum_{j\in\mathscr{S}_t} n_j} \theta_k^{t+1} \tag{1}$$

where θ^{t+1} is the global prototype of the following communication round and θ_k^{t+1} is the new model of the client k. Although FL avoids centralizing raw data, it still faces challenges such as non-IID data, device heterogeneity, high communication overhead, and security risks [16].

2.2. LLM fine-tuning

LLMs are typically pre-trained on large-scale general-purpose corpora and then adapted to downstream tasks through fine-tuning on domain-specific data [17]. The most direct strategy is full-parameter fine-tuning, where all model parameters are updated during training. To reduce the computation and communication costs, PEFT methods update only a small part of selected or additional parameters while keeping most pre-trained weights frozen. A representative PEFT method is LoRA [9], which freezes a pre-trained weight matrix $W_0 \in \mathbb{R}^{d\times n}$ and parameterizes its update with two low-rank matrices:

$$W = W_0 + \Delta W = W_0 + BA \tag{2}$$

where $A \in \mathbb{R}^{r\times n}$, $B \in \mathbb{R}^{d\times r}$ and the rank $r \ll \min(d, n)$. These fine-tuning strategies provide the methodological basis for FedLLM fine-tuning under decentralized and resource-constrained client environments.

2.3. Federated LLM fine-tuning

Building on FL and LLM fine-tuning, FedLLM fine-tuning adapts LLMs with distributed client data by performing local fine-tuning on each client and aggregating the trainable parameter updates on the server [7]. In early studies such as FedIT [6], each client only trains and transmits its local LoRA matrices, and FedAvg can be directly applied to the LoRA factors:

$$B = \frac{1}{N}\sum_{k=1}^{N} B_k, \quad A = \frac{1}{N}\sum_{k=1}^{N} A_k \tag{3}$$

where N denotes the number of participating clients, and B_k and A_k are the LoRA matrices trained by client k. This design reduces communication overhead by transmitting lightweight adaptation parameters, but it also introduces new challenges, including aggregation bias, resource heterogeneity, and data heterogeneity. Meanwhile, FedLLM fine-tuning also inherits privacy and security risks from FL. Based on these considerations, this survey reviews representative studies from three perspectives: efficiency, heterogeneity and personalization, and privacy and security.

DOI: 10.54254/2755-2721/2026.38681

3. Representative works

This part is a review of representative FedLLM fine-tuning approaches in terms of efficiency, heterogeneity and personalization, privacy and security, outlining their major designs, strengths and weaknesses.

3.1. Efficiency

FedLLM fine-tuning is a client-side computation and memory optimization that aims at minimizing the amount of work to be done by the clients and communication between the clients and server.

3.1.1. Client-side fine-tuning

The first step to achieve good performance of the federated training is to reduce the cost of local adaptation. LoRA is used in early works like FedIT [6] where it trains and shares lightweight adapters and freezes the pretrained backbone. Nevertheless, even though there are fewer parameters to train, the cost of running the large backbone cannot be removed. This prompts the choice of where local training should take place. LEFF [18] also integrates the concept of layer selection based on client capacity with that of importance-based sampling, whereby only those layers that are influential are updated and the weaker clients are assigned less workload. The focus is thus not on using lightweight adapters, but rather on distributing the effort of local training as per the computation requirements and contribution of each layer.

These are methods that limit the training elements but not backpropagation. In case of a bottleneck in storing intermediate activations, it is possible to modify how updates are performed. FedKSeed [10] uses a finite set of shared random seeds with zeroth-order optimization. For a perturbation generated by seed s_j, the client computes a scalar estimate through two forward loss evaluations:

$$\hat{g}_j = \frac{\mathcal{L}(w_i+\epsilon z_j,x)-\mathcal{L}(w_i-\epsilon z_j,x)}{2\epsilon} \tag{4}$$

where w_i denotes client i's model parameters, x is a local sample, $\mathcal{L}$ is the training loss, z_j is the seed-generated perturbation, andϵcontrols its scale. Since the perturbation is able to be regenerated using its seed, clients exchange seeds and scalar estimates rather than complete parameter updates. The server stores these estimates by seed, so that clients can rebuild the global model without replaying an ever-increasing update history. This relates local optimization to memory-efficient communication and compactness, but the quality of updates relies on the directions in which the perturbations are sampled.

Compact updates may be supported by the use of shared randomness without finite-difference estimation. More recently, FedKRSO [19] estimates projected first-order gradients in random subspaces of commonality and transfers the respective accumulators of updates. The FedKSeed approach is to represent the updates as scalar approximations in the direction of perturbation, whereas FedKRSO adds up gradient-based updates in a low-dimensional subspace. Both methods are thus similar in that they both serve the same efficiency purposes of sharing randomness but differ in terms of how local update information is collected.

DOI: 10.54254/2755-2721/2026.38681

3.1.2. Client-server communication

The communication can also be minimized without compromising the dense local adapter training. FLASC [20], instead of limiting the LoRA parameters that clients can optimize, uses sparsity only when exchanging parameters. The client trains the entire adapter locally and sends some entries, with independently configurable upload and download sparsity. This allows asymmetric bandwidth but at the same time maintains the ability to adapt locally. Its main difference is that the parameters not communicated are not excluded from training forever. Thus, the savings on communications do not require sparse local optimization.

One of the other approaches is to make the representation that can be trained itself more compact. Instead of sparsifying conventional LoRA factors, Fed-SB [21] shares and freezes the outer matrices A and B, enabling clients to train only a small middle matrix R_i. With equal client weights, the server aggregates:

$$R^{agg} = \frac{1}{c}\sum_{i=1}^{c} R_i, \quad \Delta W^{agg} = B\left(\frac{1}{c}\sum_{i=1}^{c} R_i\right)A \tag{5}$$

where c is the number of participating clients, $R_i \in \mathbb{R}^{r\times r}$ is the trainable matrix with LoRA rank r, and R^{agg} and ΔW^{agg} are the aggregate middle matrix and weight change. The outer factors are shared, which implies that when averaging the middle matrices, it is also equivalent to taking the average of the respective weight updates. Therefore, while FLASC minimizes communication by communicating selectively, Fed-SB reduces both the number of trainable parameters and communication cost with its adapter structure. It requires a fixed set of shared bases on which the efficiency relies, therefore the initialization of these bases is a significant design decision.

3.1.3. Discussion

The efficiency-based approaches have been applied to minimize the cost of FedLLM fine-tuning in various directions, such as local training parameters, optimization memory and client-server communication. Such designs are capable of decentralizing LLM adaptation when there is a scarcity of resources on the clients. Nevertheless, it can be expensive on one dimension and cheap on the other. As an example, reducing the number of transmitted parameters can still require running the large backbone locally, but compactness of communication can lead to more reconstruction or server-side calculation. Future techniques ought to measure efficiency with greater depth and take into account the adaptability of design that balances computation, communication and model quality within realistic limitations of the clients.

3.2. Heterogeneity and personalization

The FedLLM fine-tuning should be able to handle the variations in resources of clients and local tasks. These differences inspire two complementary directions, which support heterogeneous adaptation configurations and knowledge sharing organization to satisfy client-specific requirements.

3.2.1. Heterogeneous configuration and aggregation

Different resource budgets motivate different LoRA ranks, but unequal ranks prevent direct averaging of adapter factors. Even with equal ranks, averaging factors separately generally differs

DOI: 10.54254/2755-2721/2026.38681

from averaging their products. Aggregation design therefore needs to address both dimensional compatibility and aggregation accuracy.

FLoRA [11] solves the two issues by combining factors on the side of clients. It is weighted aggregation in k participating clients:

$$A = p_1 A_1 \oplus p_2 A_2 \oplus \cdots \oplus p_k A_k, \quad B = B_1 \oplus B_2 \oplus \cdots \oplus B_k \tag{6}$$

where A_k and B_k are client k's LoRA factors, p_k is the normalized aggregation weight, and $\oplus$ concatenates A_k vertically and B_k horizontally. The global factors A and B preserve the weighted sum of local updates while accommodating different ranks. However, the redistributed adapter size grows with the sum of participating ranks.

Instead of retaining stacked factors, FlexLoRA [22] aggregates full-size update matrices and uses truncated singular value decomposition to redistribute updates according to client-specific ranks:

$$W_g = \sum_{k=1}^{K} p_k s B_k A_k = U \Sigma V^{\top}, \quad W_{g,k} = U_{r_k} \Sigma_{r_k} V_{r_k}^{\top} \approx W_g \tag{7}$$

where W_g is the aggregated LoRA update, s is the LoRA scaling factor, and U, Σ, V are its SVD components. The subscript r_k denotes retention of the leading r_k singular components, producing the rank-r_k approximation $W_{g,k}$ for client k. Unlike stacking, this approach bounds redistributed adapter ranks, but introduces decomposition overhead and potential information loss through truncation.

FSLoRA [23] allows one to reconstruct and decompose updates, which is a new way of training the global LoRA submatrices selectively. This allows partial updates in a shared representation with unequal resource budgets and its analysis also relates sketching ratios to convergence behavior.

3.2.2. Client-specific personalization

Compatible aggregation does not guarantee that the shared model is suitable for all data and tasks of a client. Personalization, in turn, involves what information to share, how it should be combined with local knowledge and who needs to work together.

At the matrix level, FedSA-LoRA [24] aggregates A while retaining B locally. The selective aggregation retains local specialization in a partially shared adapter because it finds that the A factors are relatively transferable and the B factors capture client-specific knowledge, respectively.

Going to module-level separation, FedALT [4] preserves an Individual LoRA between rounds and adds a Rest-of-World (RoW) LoRA that is aggregated using other clients. The Individual LoRA and personal mixer are updated during local training, but the RoW LoRA is fixed. Their contributions are merged at an adapted linear layer by:

$$y = W_0 x + \alpha_k(x) B_k^L A_k^L x + (1 - \alpha_k(x)) B_k^R A_k^R x \tag{8}$$

where W_0 is the frozen pre-trained weight, x and y are the layer input and output, and L and R identify the Individual and RoW factors of client k. The mixing weight $\alpha_k(x)$ adjusts their contributions for each input, allowing shared knowledge to complement rather than overwrite local adaptation.

In addition to distinguishing between local and shared parts, FedTreeLoRA [25] builds a tree-like aggregation hierarchy based on the similarity of clients. It encourages wider sharing in shallow

DOI: 10.54254/2755-2721/2026.38681

layers and more specialized collaboration in deeper layers, extending personalization from deciding what to share to deciding with whom and at which depth to share it.

3.2.3. Discussion

The heterogeneity and personalization techniques that are currently available are helpful in the fine-tuning of FedLLM as they can provide unequal adaptation settings and retain the knowledge of each client. However, most of them have been interested in a rather static client scenario, but in practice, there is always some change in data, task, preferences, or participation pattern of clients. The next step should be to focus on future research of long-term customization, retention of knowledge, and changes in the conditions of clients, not just how the average performance is measured after one training procedure.

3.3. Privacy and security

Local storage of raw data does not prevent privacy leakage or malicious manipulation. Privacy protection restricts what an adversary can learn about the client data, while security defenses aim to maintain training integrity and safe model behavior.

3.3.1. Privacy leakage and protection

The exchanged parameters are also able to disclose the information about local examples. Vu et al. [26] demonstrate that active membership inference attacks can determine whether a particular example was used in the training of a client locally, which implies that parameter-efficient sharing is not sufficient to guarantee data privacy.

In order to restrict such leakage, DP-LoRA [27] employs gradient clipping and Gaussian perturbation on both LoRA factors in the process of federated fine-tuning. Taking A_k as an example, the procedure is:

$$g_{A_k} \leftarrow \frac{g_{A_k}}{max\left(\left(1, \|g_{A_k}\|_2 / C\right)\right)}, \quad g_{A_k} \leftarrow g_{A_k} + \mathcal{N}\left(0, \sigma^2 C^2 I\right) \tag{9}$$

where g_{A_k} is the local gradient, C is the clipping threshold, σ controls the noise scale, and I is the identity matrix. The noise calibration and privacy accounting are used to decide on the protection in training steps.

PPFedIT [12] does not use gradient perturbation but instead it modifies the way in which the private data are used to affect the shared parameters. It produces and filters synthetic instructions, trains two different sets of parameters on the examples of the private and the synthetic, and then combines them at the server side. This architecture minimizes the training-data extraction during the few-shot experiments that were tested as well as enhances the model utility.

3.3.2. Malicious participant attacks and defenses

Malicious clients are able to attack a shared model in the sense of not reducing performance of good tasks. ShadeEdit [28] provides an example of this by performing knowledge editing on a specific target, and it is also possible to use paraphrase-based edit-data selection and adaptive update clipping as tools that enhance malicious edits and prevent them from being mitigated by aggregation techniques.

DOI: 10.54254/2755-2721/2026.38681

To restrict the malicious influence at training time, FedByZO [29] uses robust aggregation (trimmed mean or Krum) for low-dimensional zeroth-order estimates. For K clients and H local steps per round, the server does:

$$R_{t,l} = \mathscr{R}\left(\{g_{k,t,l}\}_{k=1}^{K}\right), \quad w^{t+1} = w^{t} - \eta \sum_{l=1}^{H} Z_{t,l} R_{t,l} \tag{10}$$

where $g_{k,t,l}$ is the directional-estimate vector of client k at local step l in round t, $Z_{t,l}$ has common random perturbation directions, and R is a robust aggregation rule producing $R_{t,l}$. The global parameters w^{t} are updated with learning rate η. This mechanism combines directional estimates in a robust manner prior to the reconstruction of full-dimensional model updates.

3.3.3. Discussion

Privacy-oriented techniques will minimize the loss of information in updates and security-oriented strategies are to reduce the impact of malicious interference in aggregation. But these objectives are not mutually exclusive. The safety of training data is not a guarantee that it is resistant to malicious updates, and strong aggregation does not mean the confidentiality of the data or safe model behavior. It would be important to assess privacy leakage, attack success, safety deterioration, performance of benign tasks, and overhead of protection in future research with the threat models being well defined.

4. Datasets and evaluation benchmarks

This part examines the representative FedLLM fine-tuning datasets and evaluation benchmarks, focusing on their task coverage, client partitioning, and assessment criteria.

4.1. Datasets

The selection of datasets in FedLLM fine-tuning must be representative of the tasks that are to be adapted and client differences. General instruction tuning, cross-task generalization, multilingual adaptation, and conversational interaction are supported by representative datasets.

In the case of general instruction tuning, Alpaca [30] offers instruction-response pairs generated by a model, and Dolly-15K [31] has human-written examples on question answering, summarization, and information extraction. To adapt to cross-task scenarios, Super-NaturalInstructions [32] offers various tasks that are defined in natural language. It has task metadata which allows task-based client construction to study heterogeneous adaptation and generalization to new tasks. These resources were not initially developed as federated datasets but they can be redistributed among clients based on random or task partitioning.

Besides the generic resources, Fed-Aya and Fed-WildChat of FedLLM-Bench [8] are designed for perform multi-language instruction training and multi-turn dialogue. The clients that using these two datasets are designed in such a way that they use annotator IDs or IP addresses as the basis of their construction with the differences between them being preserved based on the sources of data used. These groupings serve as an alternative to the re-distribution of centralized datasets, but the source identifiers are not necessarily individual users or physical devices. Table 1 shows language coverage and partitioning schemes which can be representative of the experiments which have been carried out separately.

DOI: 10.54254/2755-2721/2026.38681

Table 1. Some representative FedLLM fine-tuning datasets

Dataset	Data Organization	Language	Main Purpose	Client Partitioning
Alpaca	Artificial partitioning	English	General instruction following	Random partitioning
Dolly-15K	Artificial partitioning	English	General instruction following	Dirichlet sampling over task categories
Super-NaturalInstructions	Artificial partitioning	Multilingual	Cross-task instruction learning	Grouping by task ID
Fed-Aya	Source-based grouping	Multilingual	Multilingual instruction following	Grouping by annotator ID
Fed-WildChat	Source-based grouping	Multilingual	Multi-turn conversation	Grouping by IP address

4.2. Evaluation benchmarks

The evaluation criteria must be based on the capabilities that are being fine-tuned as well as whether other more general model capabilities remain intact. Table 2 is a summary of representative benchmarks in general knowledge, mathematical reasoning, code generation and conversational instruction following.

In the General category, MMLU [33] tests knowledge and comprehension over 57 subjects, and MT-Bench [34] supplements this evaluation by testing multi-turn instruction following with GPT-4 scoring.

In the case of mathematical reasoning, MathQA [35] is used to test math word-problem solving using operation-based formalisms and MATH [36] extends the testing to more difficult competition-level problems. These benchmarks primarily determine whether the model can generate accurate responses by means of mathematical reasoning.

In terms of code generation, APPS [37] considers the task of producing Python programs based on natural language descriptions of problems, whereas DS-1000 [38] is a benchmark of real data science coding problems with the use of popular Python libraries. Both measure the performance of generated programs in terms of test cases and they are primarily measured by the pass rate.

Together, these benchmarks cover specialized task performance and wider model capabilities. The comparison of results before and after fine-tuning allows identifying the gains in adaptation as well as possible degradation of capabilities. These comparisons should be conducted under the same evaluation conditions and take into account both correctness of tasks and quality of conversations.

DOI: 10.54254/2755-2721/2026.38681

Table 2. Representative FedLLM fine-tuning evaluation benchmarks

Benchmark	Domain	Evaluation Objective	Main Evaluation Criteria
MMLU	General	Multitask understanding across 57 subjects	Accuracy
MT-Bench	General	Multi-turn conversation and instruction following	GPT-4 Score
MathQA	Mathematics	Math word-problem solving with operation-based formalisms	Accuracy
MATH	Mathematics	Competition-level mathematical problem solving	Accuracy
APPS	Code	Python code generation from natural-language specifications	Pass Rate
DS-1000	Code	Data science code generation with Python libraries	Pass Rate

5. Key challenges and future research prospects

This section discusses with some challenges in FedLLM fine-tuning and the future work.

5.1. Resource-adaptive large-small model collaboration

The issue of the limitation of resources is an inherent obstacle to fine-tuning FedLLM. Though it can be parameter-efficient, even then a large frozen backbone should be stored and run by the clients. The disparity in terms of memory, bandwidth, and accessibility makes a one-size-fits-all training arrangement unsuitable for all participants. Heterogeneous adapter rank allows flexibility but does not allow full cooperation between the clients that are based on different backbones.

Large-small model collaboration is a promising approach as it enables lightweight client models to learn on private data and share knowledge with a server-side LLM. FedMKT [39] demonstrates this possibility by selective mutual knowledge transfer and token alignment. However, the capacity and domain expertise of the models may restrict the transferable knowledge, and auxiliary data used for distillation may not represent specialized client tasks. An open question is how to find useful knowledge without exposing private examples or allowing stronger models to suppress client-specific expertise.

Future studies ought to create resource-adaptive collaboration that determines what knowledge to share, which parts to update and when it is necessary to communicate. These decisions must be sensitive to the dynamic nature of device conditions but still maintain effective transfer of knowledge across heterogeneous models. Evaluation must also identify real efficiency gains and not just shifting computation to the server. The comparison of peak memory, total communication and elapsed training time should be done at equivalent model quality.

DOI: 10.54254/2755-2721/2026.38681

5.2. Split federated fine-tuning

Whereas large-small model collaboration minimizes the client load by altering the manner of knowledge exchange among heterogeneous models, split federated fine-tuning follows another path by modifying where LLM computation is carried out. Rather than making each client hold and operate the entire LLM backbone, the model can be divided between clients and the server. Clients perform local layers and send intermediate hidden states whereas the server performs the other layers and sends gradients back to the clients for local updates. This concept has started being applied to LLMs in recent studies. FedSEA-LLaMA [40] proposes a framework of federated splitting of LLaMA2 with dynamic partition points, attention-mask compression.

Nevertheless, split FedLLM fine-tuning also poses new challenges. The communication can be frequent and latency-sensitive as hidden states and gradients are required to be exchanged across the split point, especially in autoregressive LLMs. Split learning does not necessarily address privacy risks because intermediate representations may leak private information. Besides, the selection of the partition point influences client cost, server burden, communication overhead, and model utility.

Future research should develop adaptive partitioning strategies that jointly consider device resources, network conditions, privacy leakage, and downstream task requirements. It is also important to reduce the communication cost of hidden-state exchange and design privacy-preserving mechanisms for intermediate representations. This direction is particularly relevant for edge environments where clients cannot locally maintain large LLM backbones.

5.3. Continual fine-tuning and lifelong personalization

The data, tasks and preferences of clients change with time and this makes it hard to retain knowledge and personalize the model for them. The new local updates can disrupt the previous capabilities and irregular attendance results in different levels of adaptation of the clients. FedLLM fine-tuning therefore needs to deal with not only heterogeneous clients but also constantly changing client knowledge.

Continual fine-tuning and lifelong personalization offer a natural avenue, as also noted by Wu et al. [2]. Modular adaptation may be generalized to selectively reuse, update, and consolidate shared and private knowledge in light of shifting tasks. Yet maintaining a separate adapter for each task would incur growing storage and maintenance costs. Thus future methods ought to balance adaptation capability with bounded memory, e.g., through selective consolidation/local distillation to retain useful abilities without ever-growing models.

Aggregation must also take into account relevance and recency of client updates without automatically weakening infrequently observed domains. This needs to be assessed beyond static capability scores. FedLLM-Bench [8] can capture realistic client diversity, which could aid evolving task-stream evaluation. Future protocols should further quantify forgetting, adaptation speed, and persistent client-level benefits under changing participation. These capabilities may also support personalized LLM agents that learn from private interaction trajectories through federated updates.

5.4. Privacy-preserving and safety-aligned fine-tuning

The training-data privacy preservation does not make a fine-tuned LLM safe, as the resistance to malicious updates does not guarantee data confidentiality. Safety alignment can be compromised even with multiple existing federated defenses through malicious local fine-tuning. The main

DOI: 10.54254/2755-2721/2026.38681

problem is thus how to secure personal information without allowing local adaptation to create undesirable global behavior.

This is further complicated by the lack of access to client information. Servers are not able to routinely check on personal examples in order to differentiate between valid specialization and malicious manipulation. Secure aggregation can also limit defenses which examine single updates, but differential privacy does not stop unsafe outputs itself. The interaction between privacy, robustness and safety goals should thus be considered when developing protection mechanisms.

Safety preservation needs to be integrated in future research into local adaptation and server-side validation. Possible solutions are limiting negative changes of behavior, testing aggregated models on reliable safety tasks, and using specific correction in the event of a decline in safety. Simultaneously, there is a need to ensure that legitimate personalization is not compromised and that benign requests are not excessively rejected. The assessment must jointly analyze privacy promises, empirical leakage, attack efficacy, retention of safety, performance of benign tasks, and protection overhead after repeated federated fine-tuning.

6. Conclusion

FedLLM fine-tuning is a promising paradigm of the adaptation of large language models to decentralized and privacy-sensitive data. This direction has been motivated in the introduction as an attempt to address the increasing demand to employ local domain-specific and user-specific information without directly centralizing the examples. Hence, it offers a natural link between LLM adaptation and federated learning.

The survey has covered the background, typical methods, datasets and benchmarks of FedLLM fine-tuning as well as the future trends. To be more precise, this paper categorized the literature on the basis of three significant technical aspects including efficiency, heterogeneity and personalization, and privacy and security. This paper has also presented some of the most common datasets and benchmarks in order to understand the way in which existing techniques are tested with various tasks and clients.

In this review, it can be observed that FedLLM fine-tuning is no longer an easy extension of traditional federated learning. Recent research has created specialized mechanisms for parameter-efficient training, heterogeneous aggregation, personalized adaptation, privacy protection, and robust defense. At the same time, evaluation is also becoming more systematic, moving beyond task accuracy toward resource cost, client partitioning, personalization benefit, and capability preservation.

Although this has been achieved, it is still difficult to implement in practice. The resource-adaptive collaboration, split federated fine-tuning, continual personalization and privacy-preserving safety alignment are several promising directions for future research. These directions are significant in rendering FedLLM fine-tuning more efficient, reliable, and relevant to realistic decentralized settings.

References

[1] Liu, H., Li, C., Wu, Q., et al. (2023). Visual instruction tuning. *Advances in Neural Information Processing Systems, 36*, 34892–34916.

[2] Wu, Y., Tian, C., Li, J., et al. (2026). A survey on federated fine-tuning of large language models. *Transactions on Machine Learning Research*.

[3] Woisetschläger, H., Erben, A., Wang, S., Mayer, R., & Jacobsen, H.-A. (2024). A survey on efficient federated learning methods for foundation model training. *Proceedings of the International Joint Conference on Artificial*

DOI: 10.54254/2755-2721/2026.38681

Intelligence (IJCAI), 8317–8325.
[4] Bian, J., Wang, L., Zhang, L., et al. (2026). FedALT: Federated fine-tuning through adaptive local training with Rest-of-World LoRA. *Proceedings of the AAAI Conference on Artificial Intelligence, 40*(24), 19728–19736.
[5] Yao, Y., Zhang, J., Wu, J., et al. (2026). Federated large language models: Current progress and future directions. *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, 203–221.
[6] Zhang, J., Vahidian, S., Kuo, M., et al. (2024). Towards building the FederatedGPT: Federated instruction tuning. *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 6915–6919.
[7] Ye, R., Wang, W., Chai, J., et al. (2024). OpenFedLLM: Training large language models on decentralized private data via federated learning. *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 6137-6147.
[8] Ye, R., Ge, R., Zhu, X., et al. (2024). FedLLM-Bench: Realistic benchmarks for federated learning of large language models. *Advances in Neural Information Processing Systems, 37*, 111106-111130.
[9] Hu, E. J., Shen, Y., Wallis, P., et al. (2022). LoRA: Low-rank adaptation of large language models. *Proceedings of the 10th International Conference on Learning Representations (ICLR)*.
[10] Qin, Z., Chen, D., Qian, B., et al. (2024). Federated full-parameter tuning of billion-sized language models with communication cost under 18 kilobytes. *Proceedings of the 41st International Conference on Machine Learning (ICML)*, 41473-41497.
[11] Wang, Z., Shen, Z., He, Y., et al. (2024). FLoRA: Federated fine-tuning large language models with heterogeneous low-rank adaptations. *Advances in Neural Information Processing Systems, 37*, 22513-22533.
[12] Zhang, Z., Zhang, J., Huang, J., et al. (2026). PPFedIT: Towards privacy-preserving federated instruction tuning with few-shot local examples. *ACM Transactions on Intelligent Systems and Technology, 17*(5), 1-23.
[13] Yang, Y., Long, G., Lu, Q., et al. (2025). Federated low-rank adaptation for foundation models: A survey. *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence (IJCAI)*, 10779-10787.
[14] Kairouz, P., & McMahan, H. B. (2021). Advances and open problems in federated learning. *Foundations and Trends in Machine Learning, 14*(1-2), 1-210.
[15] McMahan, B., Moore, E., Ramage, D., et al. (2017). Communication-efficient learning of deep networks from decentralized data. *Artificial Intelligence and Statistics*, 1273-1282.
[16] Li, T., Sahu, A. K., Talwalkar, A., & Smith, V. (2020). Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine, 37*(3), 50-60.
[17] Raffel, C., Shazeer, N., Roberts, A., et al. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research, 21*(140), 1-67.
[18] Shen, J., Cheng, N., Xu, W., Wang, H., Guo, Y., & Xu, J. (2025). LLM at network edge: A layer-wise efficient federated fine-tuning approach. *Advances in Neural Information Processing Systems, 38*.
[19] Yang, G., Wu, T., Guo, Y., et al. (2026). FedKRSO: Communication and memory efficient federated fine-tuning of large language models. *IEEE INFOCOM 2026 - IEEE Conference on Computer Communications*, 1–10.
[20] Kuo, K., Raje, A., Rajesh, K., & Smith, V. (2026). FLASC: Federated LoRA with sparse communication. *Proceedings of the ACM Conference on AI and Agentic Systems*, 990–1008.
[21] Singhal, R., Ponkshe, K., Vartak, R., et al. (2026). Fed-SB: A silver bullet for extreme communication efficiency and performance in (private) federated LoRA fine-tuning. *Transactions on Machine Learning Research*.
[22] Bai, J., Chen, D., Qian, B., et al. (2024). Federated fine-tuning of large language models under heterogeneous tasks and client resources. *Advances in Neural Information Processing Systems, 37*, 14457–14483.
[23] Fang, W., Han, D. J., Yuan, L., et al. (2026). Federated sketching LoRA: A flexible framework for heterogeneous collaborative fine-tuning of LLMs. *Proceedings of the 43rd International Conference on Machine Learning (ICML)*.
[24] Guo, P., Zeng, S., Wang, Y., et al. (2025). Selective aggregation for low-rank adaptation in federated learning. *International Conference on Learning Representations*, 99003–99027.
[25] Bian, J., Wang, L., Zhang, L., et al. (2026). FedTreeLoRA: Reconciling statistical and functional heterogeneity in federated LoRA fine-tuning. *Proceedings of the 43rd International Conference on Machine Learning (ICML)*.
[26] Vu, M., Nguyen, T., & Thai, M. T. (2024). Analysis of privacy leakage in federated large language models. *International Conference on Artificial Intelligence and Statistics*, 1423-1431.
[27] Liu, X. Y., Zhu, R., Zha, D., et al. (2025). Differentially private low-rank adaptation of large language model using federated learning. *ACM Transactions on Management Information Systems, 16*(2), 1-24.
[28] Zhang, X., Liu, H., Guo, S., et al. (2026). ShadeEdit: A utility-preserving and defense-evasive knowledge manipulation attack in federated LLMs. *Proceedings of the AAAI Conference on Artificial Intelligence, 40*(41), 34845-34853.

DOI: 10.54254/2755-2721/2026.38681

[29] Egger, M., Bakshi, M., & Bitar, R. (2025). Byzantine-resilient zero-order optimization for scalable federated fine-tuning of large language models. *ICML Workshop on Efficient Systems for Foundation Models III*.
[30] Taori, R., Gulrajani, I., Zhang, T., et al. (2023). Alpaca: A strong, replicable instruction-following model. *Stanford Center for Research on Foundation Models*. Retrieved from https://crfm.stanford.edu/2023/03/13/alpaca.html
[31] Conover, M., Hayes, M., Mathur, A., et al. (2023). Free Dolly: Introducing the world's first truly open instruction-tuned LLM.
[32] Wang, Y., Mishra, S., Alipoormolabashi, P., et al. (2022). Super-NaturalInstructions: Generalization via declarative instructions on 1600+ NLP tasks. *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, 5085-5109.
[33] Hendrycks, D., Burns, C., Basart, S., et al. (2021). Measuring massive multitask language understanding. *Proceedings of the 9th International Conference on Learning Representations (ICLR)*.
[34] Zheng, L., Chiang, W. L., Sheng, Y., et al. (2023). Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. *Advances in Neural Information Processing Systems, 36*, 46595-46623.
[35] Amini, A., Gabriel, S., Lin, S., et al. (2019). MathQA: Towards interpretable math word problem solving with operation-based formalisms. *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2357-2367.
[36] Hendrycks, D., Burns, C., Kadavath, S., et al. (2021). Measuring mathematical problem solving with the MATH dataset. *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks, 1*.
[37] Hendrycks, D., Basart, S., Kadavath, S., et al. (2021). Measuring coding challenge competence with APPS. *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks, 1*.
[38] Lai, Y., Li, C., Wang, Y., et al. (2023). DS-1000: A natural and reliable benchmark for data science code generation. *International Conference on Machine Learning, 18319–18345*.
[39] Fan, T., Ma, G., Kang, Y., et al. (2025). FedMKT: Federated mutual knowledge transfer for large and small language models. *Proceedings of the 31st International Conference on Computational Linguistics, 243–255*.
[40] Zhang, Z., Zhang, H., Li, W., et al. (2026). FedSEA-LLaMA: A secure, efficient and adaptive federated splitting framework for large language models. *Proceedings of the AAAI Conference on Artificial Intelligence, 40*(34), 28680–28688.