FedLLM-Factory: A Unified Framework for Federated Large Language Model Fine-tuning

Boyi Liu
DS, City University of Hong Kong
SKLCCSE, Beihang University
Hong Kong, China
boy.liu@my.cityu.edu.hk

Abstract

Federated Large Language Model (FedLLM) fine-tuning enables the collaborative adaptation of powerful models to domain-specific tasks without compromising data privacy. However, deploying FedLLM in practical edge environments is severely hindered by multifaceted system constraints, including dynamic client availability and heterogeneous computing resources. Existing frameworks often lack the capability to seamlessly bridge idealized algorithmic prototyping with realistic, event-driven system simulations. To address this, we present FedLLM-Factory, a comprehensive and highly extensible open-source library and benchmark tailored for FedLLM. FedLLM-Factory introduces an innovative dual-mode architecture: a Normal mode for theoretical algorithmic validation, and an event-driven Real-world mode that accurately models practical execution constraints. Furthermore, it systematically supports complex composite heterogeneity, which includes variations in devices, states, ranks, and communications, along with flexible personalization configurations. By integrating over 15 federated fine-tuning algorithms and maintaining extensive compatibility with the Hugging Face ecosystem across multiple NLP tasks, FedLLM-Factory provides the community with a robust, unified platform for end-to-end FedLLM research and deployment. The code is in https://github.com/boyi-liu/FedLLM-Factory.

ACM Reference Format:
Boyi Liu. 2026. FedLLM-Factory: A Unified Framework for Federated Large Language Model Fine-tuning. In . ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introduction

Large Language Models (LLMs) have demonstrated unprecedented capabilities across diverse natural language processing tasks. To leverage these models on privacy-sensitive user data, Federated Learning (FL) [7] has been widely adopted to fine-tune LLMs collaboratively across distributed clients without raw data sharing (see Fig. 1). Despite the rapid development of algorithmic solutions, benchmarking Federated LLM (FedLLM) fine-tuning remains challenging due to the complex interplay between algorithmic design and real-world system constraints.

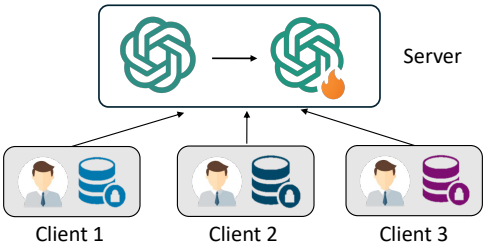

Figure 1: Federated LLM fine-tuning

Existing frameworks often evaluate algorithms under ideal synchronization assumptions or model asynchrony as simplistic staleness patterns. They struggle to capture the complex, multifaceted heterogeneity inherent in real-world setups—such as clients randomly dropping offline, fluctuating network bandwidths, or heterogeneous hardware requiring distinct fine-tuning ranks (*e.g.*, in LoRA [3]). Consequently, there is a critical need for a platform that allows researchers to rigorously test their FedLLM algorithms under highly realistic system conditions while maintaining ease of use.

In this work, we introduce **FedLLM-Factory**, a versatile framework designed specifically for federated LLM fine-tuning. FedLLM-Factory overcomes the limitations of existing libraries by introducing an innovative event-driven *Realistic* execution mode alongside a standard *Prototype* mode. Furthermore, it systematically models complex system heterogeneity—incorporating device, state, rank, and communication dimensions—and seamlessly integrates state-of-the-art personalization mechanisms. In summary, our main contributions are as follows:

- We present FedLLM-Factory, a comprehensive and highly extensible open-source library tailored for FedLLM fine-tuning, which seamlessly bridges idealized algorithmic prototyping with realistic system simulations.
- We introduce an innovative dual-mode architecture, comprising a Prototype mode for algorithmic prototyping and an event-driven Realistic mode for practical execution constraints.
- We provide robust algorithmic scalability by integrating over 15 baselines and ensuring broad compatibility with the Hugging Face ecosystem for diverse NLP evaluation tasks.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
Conference'17, July 2017, Washington, DC, USA
© 2026 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

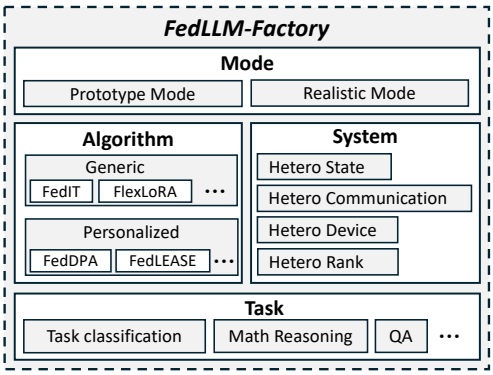

Figure 2: FedLLM-Factory framework

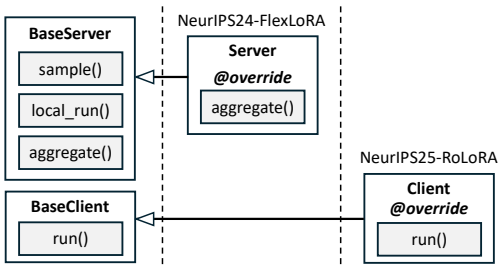

Figure 3: Example of Prototype mode

2 System Design

2.1 FedLLM-Factory Overview

FedLLM-Factory is built for scalability and ease of integration. It currently supports over 15 federated fine-tuning algorithms out of the box. The framework evaluates models across multiple NLP domains, including Text Classification, Question Answering, and Mathematical Reasoning. At the model layer, it maintains broad compatibility with the Hugging Face ecosystem, allowing researchers to effortlessly plug in various state-of-the-art LLMs.

To seamlessly bridge idealized prototyping and real-world system simulations, a core innovation of FedLLM-Factory is that it natively supports two distinct execution modes: a Prototype mode and a Realistic mode.

2.2 Prototype Mode

Principle. The Prototype mode simulates an ideal FL environment where communication is perfectly stable, and devices are perpetually online. The primary goal of this mode is to allow researchers to focus purely on *algorithmic efficacy*, enabling rapid prototyping and mathematical validation. In this setting, the framework prioritizes the rigorous implementation and testing of federated algorithms, ensuring that developers can verify theoretical designs without the interference of complex system bottlenecks or event scheduling.

Design Rationale. To facilitate rapid prototyping and significantly reduce boilerplate code, we abstract the standard behaviors of federated learning into discrete lifecycle steps through a modular, object-oriented design. Specifically, the framework provides two core foundational classes:

- **`BaseServer`:** This class manages and implements standard server-side operations, such as client sampling, model distribution, and global aggregation. Researchers can introduce new server-side algorithmic logic (*e.g.*, customized aggregation mechanisms) simply by inheriting from `BaseServer` and selectively overriding specific methods like `sample()`, `local_run()` or `aggregate()`.
- **`BaseClient`:** This class encapsulates standard client-side behaviors, handling the local execution and LLM fine-tuning processes on distributed data. To customize local training algorithms, developers only need to inherit from `BaseClient` and override essential methods `run()`.

By strictly decoupling the server and client functionalities into these extensible base classes, FedLLM-Factory allows developers to focus entirely on implementing their novel algorithmic logic without worrying about the underlying execution flow.

Example. Fig. 3 presents an example of algorithm implementation in Prototype mode. To introduce a new federated fine-tuning algorithm, researchers only need to inherit from these base classes and selectively override the necessary methods.

For instance, implementing FlexLoRA [1] simply requires creating a custom `Server` class that overrides the `aggregate()` method. Similarly, RoLoRA [2] can be developed by creating a custom `Client` class that overrides the `run()` method. This modular, object-oriented design significantly reduces boilerplate code and allows researchers to rapidly deploy new algorithmic logic.

2.3 Realistic Mode

Principle. While the Prototype mode excels at rapid algorithmic validation, it inherently operates under the assumption of an idealized environment. To address this limitation, the Realistic mode is introduced to simulate practical deployment scenarios. Specifically, the Realistic mode models actual training durations, communication bottlenecks, and dynamic training windows. The primary advantage of the Realistic mode is its ability to rigorously evaluate the robustness and true performance of FedLLM algorithms under complex, non-idealized edge environments. By seamlessly integrating realistic system bottlenecks and composite heterogeneity, it provides researchers with a highly authentic testing ground, effectively bridging the gap between theoretical FL research and practical system deployment.

Design Rationale. FedLLM-Factory employs an event-driven architecture, abstracting FL processes into discrete events:

- `WINDOW_OPEN`: Triggered when a new training round or time window initiates, allowing the server to sample available clients and distribute the global model for local training.
- `WINDOW_CLOSE`: Signals the end of the current aggregation window, prompting the server to stop accepting new client updates and proceed with the global model aggregation.
- `TRAINING_DONE`: Indicates that a selected client has successfully completed its local LLM fine-tuning computation and is ready to transition to the uploading phase.

- UPLOAD_DONE: Occurs when a client finishes transmitting its local model updates over the simulated network, successfully delivering its contribution to the server for aggregation.

Client states are rigorously modeled as IDLE, AVAILABLE, TRAINING, or UPLOADING. The server utilizes a polling handler to continuously extract and process events from a priority queue sorted by event wall clock time. This design elegantly accommodates synchronous, asynchronous, and sequential training paradigms under a unified architecture.

Fig. 4 illustrates the core event handling loop of the Realistic mode. The system initializes WINDOW_OPEN and WINDOW_CLOSE events based on a predefined state log and pushes them into a central event queue. The server utilizes a polling handler to continuously extract and process events from this queue. The execution logic flows as follows:

- When the handler extracts a WINDOW_OPEN event, it attempts to initiate training on selected clients. Upon successful completion of the local computation, a TRAINING_DONE event is inserted back into the queue.
- Extracting a TRAINING_DONE event triggers the *Upload* process. Once the simulated network transmission finishes, an UPLOAD_DONE event is inserted into the queue.
- Upon handling an UPLOAD_DONE event, the server checks if sufficient client uploads have been collected to optionally trigger a global model *Aggregation*.
- Finally, handling a WINDOW_CLOSE event triggers an update to the overall client status logs.

This continuous queue-handler mechanism elegantly accommodates synchronous, asynchronous, and sequential training paradigms under a unified architecture.

System Heterogeneity Configuration. Real-world federated LLM fine-tuning must also handle massive discrepancies across edge devices. Consequently, FedLLM-Factory supports composite heterogeneity:

- **Device and Communication Heterogeneity:** Researchers can simulate realistic uplink/downlink speeds and computational profiles (*e.g.*, Wi-Fi, 4G, 5G) [4].
- **State Heterogeneity:** Clients exhibit dynamic availability, allowing researchers to simulate probabilistic device dropouts or intermittent connectivity [6].
- **Rank Heterogeneity:** To accommodate varying hardware capacities, FedLLM-Factory supports rank heterogeneity, allowing different clients to hold fewer or more LoRA ranks dynamically [1].

These parameters can be centrally and intuitively configured via a single `sys.yaml` file, streamlining the experimental setup.

2.4 Discussions

Interplay of Two Modes. It is important to note that the Prototype mode and the Realistic mode are not entirely isolated from one another. While designed for ideal conditions, the Prototype mode can lightly simulate certain constraints by leveraging the device and rank heterogeneity features native to the Realistic mode. Conversely, the core algorithmic implementations developed and validated within the Prototype mode can be seamlessly deployed directly into the event-driven architecture of the Realistic mode.

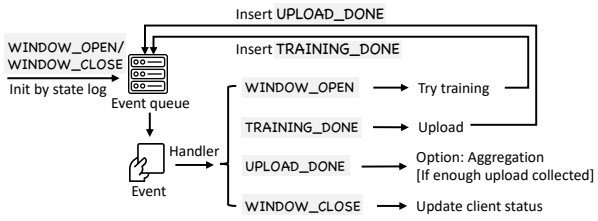

Figure 4: Event handling loop of Realistic mode.

Other Features. Building upon this interconnected foundation, practical FedLLM often requires advanced client-specific model adaptations. To address these needs, FedLLM-Factory provides robust built-in support for personalization. Specifically, we have implemented two mainstream personalization paradigms including *Model-based Personalization* [8] and *Collaboration-based Personalization* [5]. By leveraging these out-of-the-box implementations, researchers can not only toggle advanced personalization configurations with minimal code changes but also effortlessly use them as foundational baselines to develop and integrate their own customized personalization methods.

3 Conclusion

FedLLM-Factory is an open-source, versatile library and benchmark tailored for the practical advancement of federated LLM fine-tuning. By introducing an event-driven Realistic mode, complex composite heterogeneity (spanning device, communication, rank, and state), and advanced personalization support, it faithfully bridges the gap between theoretical FL research and real-world system constraints. With extensive algorithm and multi-task support, FedLLM-Factory provides the community with a robust foundation for developing and rigorously evaluating next-generation FedLLM technologies.

References

[1] Jiamu Bai, Daoyuan Chen, Bingchen Qian, Liuyi Yao, and Yaliang Li. 2024. Federated fine-tuning of large language models under heterogeneous tasks and client resources. *NeurIPS* 37 (2024), 14457–14483.
[2] Shuangyi Chen, Yuanxin Guo, Yue Ju, Hardik Dalal, Zhongwen Zhu, and Ashish J Khisti. 2025. Robust Federated Finetuning of LLMs via Alternating Optimization of LoRA. In *NeurIPS*.
[3] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Liang Wang, Weizhu Chen, et al. 2022. Lora: Low-rank adaptation of large language models. *ICLR* 1, 2 (2022), 3.
[4] Boyi Liu, Shuyuan Li, Zimu Zhou, Shuo Kang, Yiming Ma, and Yongxin Tong. 2025. Poster: Asynchronous Federated Learning Library and Benchmark with AFL-Lib. In *MobiCom*. 1281–1283.
[5] Lei Wang, Jieming Bian, Letian Zhang, and Jie Xu. 2025. Adaptive LoRA Experts Allocation and Selection for Federated Fine-Tuning. In *NeurIPS*.
[6] Chengxu Yang, Qipeng Wang, Mengwei Xu, Zhenpeng Chen, Kaigui Bian, Yunxin Liu, and Xuanzhe Liu. 2021. Characterizing impacts of heterogeneity in federated learning upon large-scale smartphone data. In *WWW*. 935–946.
[7] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. 2019. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology* 10, 2 (2019), 1–19.
[8] Yiyuan Yang, Guodong Long, Tao Shen, Jing Jiang, and Michael Blumenstein. 2024. Dual-personalizing adapter for federated foundation models. *NeurIPS* 37 (2024), 39409–39433.