FederatedScope-LLM: A Comprehensive Package for Fine-tuning Large Language Models in Federated Learning

Weirui Kuang[*]
Alibaba Group
China, Hangzhou
weirui.kwr@alibaba-inc.com

Bingchen Qian[*]
Alibaba Group
China, Hangzhou
qianbingchen.qbc@alibaba-inc.com

Zitao Li
Alibaba Group
United States, Bellevue
zitao.l@alibaba-inc.com

Daoyuan Chen
Alibaba Group
China, Hangzhou
daoyuanchen.cdy@alibaba-inc.com

Dawei Gao
Alibaba Group
China, Beijing
gaodawei.gdw@alibaba-inc.com

Xuchen Pan
Alibaba Group
China, Beijing
panxuchen.pxc@alibaba-inc.com

Yuexiang Xie
Alibaba Group
China, Hangzhou
yuexiang.xyx@alibaba-inc.com

Yaliang Li[†]
Alibaba Group
China, Hangzhou
yaliang.li@alibaba-inc.com

Bolin Ding
Alibaba Group
United States, Bellevue
bolin.ding@alibaba-inc.com

Jingren Zhou
Alibaba Group
United States, Bellevue
jingren.zhou@alibaba-inc.com

Abstract

Large language models (LLMs) have demonstrated great capabilities in various natural language understanding and generation tasks. These pre-trained LLMs can be further improved for specific downstream tasks by fine-tuning. However, the adoption of LLM in real-world applications can be hindered by privacy concerns and the resource-intensive nature of model training and fine-tuning. When multiple entities have similar interested tasks but cannot directly share their local data due to privacy regulations, federated learning (FL) is a mainstream solution to leverage the data of different entities. Besides avoiding direct data sharing, FL can also achieve rigorous data privacy protection, model intelligent property protection, and model customization via composition with different techniques. Despite the aforementioned advantages of FL, fine-tuning LLMs in FL settings still lacks adequate support from the existing frameworks and, therefore, faces challenges in optimizing the consumption of significant communication and computational resources, preparing various data for different tasks, and satisfying diverse information protection demands.

In this paper, we discuss these challenges and introduce our package **F**ederated**S**cope-**LLM** (FS-LLM) as a main contribution, which consists: (1) We build a complete end-to-end benchmarking pipeline under real-world scenarios, automizing the processes of dataset preprocessing, federated fine-tuning execution or simulation, and performance evaluation; (2) We provide comprehensive and off-the-shelf federated parameter-efficient fine-tuning (PEFT) algorithm implementations and versatile programming interfaces for future extension, enhancing the capabilities of LLMs in FL scenarios with low communication and computation costs, even without accessing the full model; (3) We adopt several accelerating and resource-efficient operators, and provide flexible pluggable sub-routines for interdisciplinary study. We conduct extensive and reproducible experiments to show the effectiveness of FS-LLM and benchmark advanced LLMs with PEFT algorithms in FL. We release FS-LLM at https://github.com/alibaba/FederatedScope/tree/llm.

CCS Concepts

• **Security and privacy → Privacy-preserving protocols**; • **Computing methodologies → Machine learning**.

Keywords

Federated Learning; Large Language Models; Benchmark

ACM Reference Format:
Weirui Kuang, Bingchen Qian, Zitao Li, Daoyuan Chen, Dawei Gao, Xuchen Pan, Yuexiang Xie, Yaliang Li, Bolin Ding, and Jingren Zhou. 2024. FederatedScope-LLM: A Comprehensive Package for Fine-tuning Large Language Models in Federated Learning. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '24), August 25–29, 2024, Barcelona, Spain.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3637528.3671573

[*] Both authors contributed equally to this research.
[†] Corresponding author.

KDD '24, August 25–29, 2024, Barcelona, Spain.

ACM ISBN 979-8-4007-0490-1/24/08
https://doi.org/10.1145/3637528.3671573

1 Introduction

Recent advances in large language models (LLMs) [5, 11, 44, 55, 60, 67, 68] have enabled a wide range of real-world applications across various domains, such as chatbots [43], writing assistants [24, 31], search engines (e.g., New Bing), tool/API retriever [46, 49] and multimodal systems [19, 23, 63]. Compared to previous pre-trained language models [17, 39], LLMs exhibit remarkable emergent abilities that have not been observed before [70]. These emergent abilities of LLMs serve as the foundation of the unprecedented proficiency and efficiency of AI systems built upon them. Consequently, both academia and industry have demonstrated a keen interest in constantly investigating the potentialities of LLMs.

When applying LLMs in practical applications, such as education, medicine, and so on, fine-tuning LLMs with domain-specific data can be essential. Fine-tuning can enrich LLMs with domain knowledge, enhance their specific ability, improve the fairness and reliability of the outputs, and prevent certain damage caused by hallucination [26]. However, fine-tuning LLMs entails a high demand for computational resources and a substantial amount of domain data that may not be sharable due to privacy concerns. The former challenge can be addressed by recent works [21, 22, 27, 32, 34, 37, 38, 47, 48], which adapt pre-trained LLMs to specific domains by tuning modules with limited trainable parameters (denoted as *adapters*). For the latter issue, one feasible solution is federated learning (FL) [2, 28, 36, 41, 50, 66], which allows multiple entities to learn model collaboratively without sharing data.

Although the existing FL frameworks [4, 54] can usually support various machine learning models, the development of federated fine-tuning on LLM is still in a premature stage because of the following **gaps** in existing work. **(i)** No existing FL package contains comprehensive and efficient implementations of LLM fine-tuning algorithms and a standardized benchmark for comparing the model performance, communication cost, and computation overhead when federated fine-tuning LLMs. **(ii)** Fine-tuning LLMs in FL is still computationally expensive on the client side, even with the parameter-efficient fine-tuning (PEFT) algorithms. **(iii)** Because pre-trained LLMs are of great intelligent property value and may not belong to clients, it might be necessary to let clients conduct federated fine-tuning without accessing the full model (e.g., closed-source LLMs). **(iv)** It is unclear whether the existing algorithms for solving advanced FL problems, such as personalized FL (pFL) [8, 58] and federated hyperparameter optimization (FedHPO) [62], are still effective with different federated fine-tuning algorithms for LLMs.

We aim to bridge the aforementioned gaps and further promote the study of fine-tuning LLMs in the context of FL. Thus, we build up a novel open-source package for fine-tuning LLMs via federated learning, called **F**ederated**S**cope-**LLM** (FS-LLM), on top of FederatedScope (FS) [65]. Our contributions can be summarized as follows:

- FS-LLM packages a collection of diverse federated fine-tuning datasets from various domains with tunable levels of heterogeneity and a suite of corresponding evaluation tasks to form a complete pipeline to benchmark federated fine-tuning LLMs algorithms in FL scenarios.
- FS-LLM provides comprehensive federated fine-tuning algorithms for LLMs with low communication and computation costs and versatile programming interfaces, which support both scenarios where clients can or cannot access the full model.
- FS-LLM is equipped with an optimized federated fine-tuning training paradigm for LLMs towards customizable efficiency-boosting (e.g., memory consumption reduction and multi-GPU parallelism) and interdisciplinary research potentials (e.g., pFL and FedHPO).
- We perform extensive experiments based on FS-LLM and investigate the empirical performances of federated fine-tuned LLMs. Based on our observations, we point out the challenges for federated fine-tuning LLMs and offer insights for future research in this emerging field.

2 Overview

In Figure 1, we illustrate the overall architecture of FS-LLM using a concrete example. As mentioned above, FS-LLM is built upon FS [65], an easy-to-use platform that provides essential modules for constructing FL courses (e.g., communication modules, aggregation strategies, and training functionality interfaces), rich extended functionality, and plenty of instantiated FL algorithms. On top of FS, we develop three enrichments to address the gaps mentioned in Sec. 1 and facilitate fine-tuning LLMs in the federated setting: **LLM-Benchmarks**, **LLM-AlgZoo**, and **LLM-Trainer**. With these modules, FS-LLM supports end-to-end federated fine-tuning for LLMs, providing (1) real-world data preparation for fine-tuning and evaluation, (2) several out-of-the-box popular fine-tuning algorithms and unified and flexible programming interfaces, (3) various accelerating and resource-efficient operators and flexible pluggable sub-routines for interdisciplinary study. We give descriptions of these modules in Sec. 3, 4, and 5, respectively. For more details of the scope of this paper, please refer to Appendix A.

3 LLM-Benchmarks: A Complete Pipeline for Benchmarking

As gap **(i)** mentioned in Sec. 1, there has yet to be a consensus in the academic community or industry on how to fairly evaluate the LLM fine-tuning algorithms in FL and what baselines are to compare. Therefore, we introduce LLM-Benchmarks, the first convenient and fair module to evaluate federated LLM fine-tuning. LLM-Benchmarks covers a complete benchmarking pipeline, consisting of stages from the construction of fine-tuning datasets to the evaluation with a collection of tasks. To facilitate replication and validation, we offer a series of look-up tables containing benchmark results for the fine-tuning datasets and their corresponding evaluation tasks. Additionally, we containerize the runtime environment of the evaluation for conveniently benchmarking the performance of the federated fine-tuned LLMs.

3.1 Federated Fine-tuning Dataset Construction

Unlike pre-training from scratch, fine-tuning LLMs is typically for adapting pre-trained LLMs to one specific domain, which can be very diverse, such as code generation and mathematical reasoning. Additionally, as considered in existing FL literature, the local datasets held by individual clients can exhibit varying degrees of heterogeneity, even within the same domain. For instance, although a set of clients share a common interest in fine-tuning an LLM for

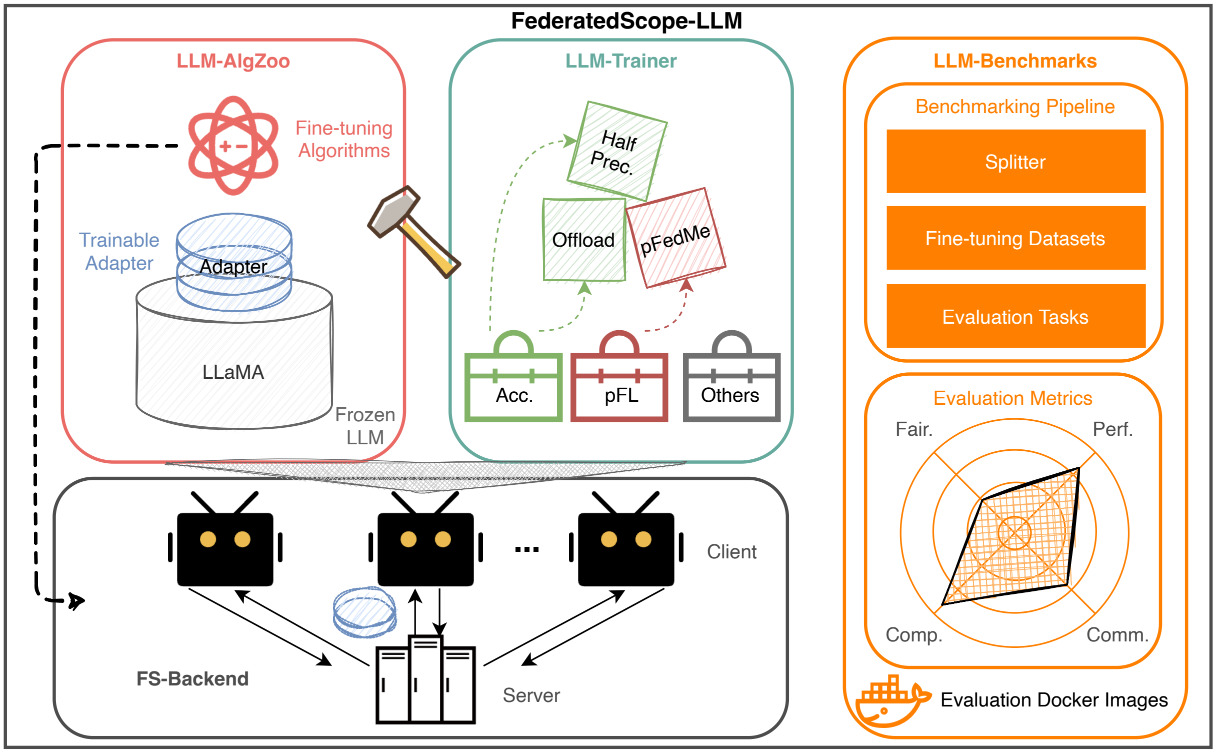

Figure 1: Overview of the architecture of FS-LLM, which consists of three main modules: LLM-Benchmarks, LLM-AlgZoo, and LLM-Trainer. As an example in the figure, we use the PEFT algorithms to fine-tune LLaMA [60] in FL, with half-precision [42] training and offloading [52] strategy and pFedMe [57] algorithm. In the figure, Acc. stands for accelerating operator, Perf. stands for performance, Comm. stands for communication, Comp. stands for computation, and Fair. stands for fairness.

code generation, the code base of some clients may mainly consist of Jave, while the code bases of other clients may contain a substantial portion of Python.

To echo the diversity of target domains and the heterogeneity of data in real-world FL scenarios, we curate three fine-tuning datasets in LLM-Benchmarks. They cover a wide range of domains and exhibit realistic data distributions across different clients. (1) *Fed-CodeAlpaca* is built from CodeAlpaca [7] to enhance LLMs’ code generation capability. It simulates a *nine*-client FL scenario. Each client’s dataset consists of coding exercises with answers limited to *one* specific programming language, such as Java or Python. (2) *Fed-Dolly* is for enhancing the capability of LLMs for generic language. We partition Databricks-dolly-15k [13] into *eight* clients’ local datasets. Each client’s dataset consists of a series of high-quality human-generated question-response pairs but is limited to *one* specific NLP task, such as information extraction or QA. (3) *Fed-GSM8K-3* is curated into *three* subsets from the train split in *GSM8K* [12], aiming to enhance the capability of LLMs for the chain of thought (CoT). Each client’s dataset consists of grade school math questions randomly partitioned from the original dataset.

It is worth noting that, in addition to the built-in datasets mentioned above, we provide splitters for partitioning the centralized dataset into a federated version based on different meta-information or with different heterogeneity degrees among clients. For example, users can apply different splitters to a centralized dataset, such as the uniform splitter, Dirichlet splitter, or meta-information splitter, to construct fine-tuning datasets that mirror the heterogeneity inherent in different FL scenarios. We provide several built-in datasets for these splitters, such as *Alpaca* [59], *cleanedAlpaca* [53], and so on. For more details about the provided fine-tuning datasets in FS-LLM, please refer to Appendix B.

3.2 Federated LLM Fine-tuning Evaluation

LLMs are known to be very powerful, but it is challenging to evaluate their capabilities by a single metric. To the best of our knowledge, there are no ready-to-use evaluation tools for assessing federated LLM fine-tuning (let alone with the personalized FL algorithms) in terms of accuracy and efficiency. To fulfill such a gap, in LLM-Benchmarks, we provide a complete benchmarking pipeline to assess LLM fine-tuning in various FL scenarios.

We argue that fine-tuning should aim to enhance one of the two aspects of LLMs: either to improve their generic language capabilities or to improve their domain-specific capabilities for one particular downstream task. Therefore, we curate three evaluation datasets from different subject areas, including *HumanEval* [10] for the code generation, *HELM* [35] for the generic language capabilities, and *GSM8K-test* [12] (the test split in *GSM8K*) for CoT. Given that different datasets employ different default evaluation metrics, for simplicity, we introduce the term *evaluation score* as a unifying descriptor for the evaluated results obtained on these datasets with their metrics. Specifically, the evaluation score represents Pass@1 score when using *HumanEval*, a mixture of metric scores on 16 subtasks when evaluating on *HELM*, and accuracy when utilizing *GSM8K-test*. And more details can be found in Appendix B.

Then, we define evaluating an LLM on a specific dataset and generating the corresponding evaluation score as an *evaluation task*. We combine each fine-tuning dataset mentioned in Sec. 3.1 with one specific evaluation task, which allows users to fairly assess the improvement of fine-tuned LLMs in FL. The fine-tuning dataset, the corresponding evaluation task, and the goal of the evaluation task are described in detail in Appendix B. Note that there is generally a distribution shift between the fine-tuning and evaluation datasets, making it much more challenging than those in other domains of FL [6, 18, 61]. To ensure the consistency of the evaluation results, we containerize the runtime environment of these evaluation tasks to docker images for conveniently assessing the performance of the federated LLM fine-tuning.

Last but not least, we also introduce a rich set of cost-related metrics to measure the efficiency of a federated fine-tuning process, including both computation costs (e.g., GPU usage, computation time, flops count) and communication costs (e.g., message size). Along with evaluation scores, these metrics could give a comprehensive assessment of the federated fine-tuning process.

4 LLM-AlgZoo: A Collection of Fine-tuning Algorithms

In addition to the benchmarking module, LLM-Benchmarks, we implement a set of popular fine-tuning algorithms in FS-LLM and introduce them as LLM-AlgZoo in this section. Aiming to fulfill the gaps **(i)** and **(ii)**, LLM-AlgZoo first includes a collection of PEFT algorithms to satisfy the constraints on the communication and computation costs in federated fine-tuning when all clients have access to the full model. However, we also realize that there are cases where the LLM owner may not be willing to share the full model in the federated fine-tuning stage. Thus, to fulfill gap **(iii)**, we further adopt a fine-tuning algorithm that does not require full model access in the FL setting. Notice that all these fine-tuning algorithms are implemented on a set of unified but flexible interfaces, which hides the underlying standard functionalities from algorithm implementors, such as coordinating communication in FL. The same interfaces can support more diverse fine-tuning algorithms in future development if users follow the same programming paradigm as our implemented algorithms.

4.1 Reducing Communication & Computation Costs with PEFT Algorithms

Achieving communication and computation efficiencies are two major challenges for fine-tuning LLMs in FL. The communication bottleneck arises from the limited bandwidth of the internet connection between the server and the clients. This challenge becomes exacerbated when full-parameter fine-tuning LLMs in FL, as they require transmitting more parameters than previous pre-trained language models. And the computation efficiency is the other critical issue for federated fine-tuning LLMs. For example, full-parameter fine-tuning LLaMA-7B requires about 28GB of GPU memory for the model. In addition, the SGD optimizer and the gradients need another 92GB of GPU memory, leading to at least 112GB of GPU memory in total - this is unaffordable for resource-limited entities.

We provide a set of implemented PEFT algorithms in FS-LLM as solutions for our users to encounter these challenges, including LoRA [22], prefix-tuning [34], P-tuning [38], and prompt tuning [32]. These algorithms perform fine-tuning by only training (additional) modules with limited parameters, known as *adapters*, but keep other parameters frozen. Compared to full-parameter fine-tuning in FL, clients only need to transmit the adapters in each communication round, which reduces the transmission time to tens of seconds or even seconds. Meanwhile, PEFT algorithms reduce the computation cost and make local training more viable for resource-limited clients. Thus, based on considerations of communication and computation costs, LLM-AlgZoo adopts PEFT algorithms and makes them considerably more viable for resource-limited entities when federated fine-tuning LLMs.

4.2 Fine-tuning without Accessing Full Model

Many existing LLMs [1, 43, 44] are closed-source for some reasons such as high training costs, preventing training data leakage, and maintaining commercial secrets. However, in some commercial scenarios, downstream customers may not be satisfied with simply using APIs to perform inference on these all-around LLMs but also want to customize them more to their domain-specific data. These domain-specific data are often private, limited, and incomplete, which leads to insufficient adaptation and generalization of LLMs to the customers' needs.

To satisfy such practical demand, we adapt a privacy-preserving fine-tuning algorithm, offsite-tuning [64], to a federated version, and name it FedOT for short. In each communication round, the server will send parameters of the Adapter of the raw model (e.g., the first and the last two layers) to clients. The clients are tasked with fine-tuning the Adapter alongside the frozen emulator using their private data and sending back the tuned Adapter. At the server's end, an aggregation of all the fine-tuned Adapters takes place for progression to the subsequent round of communication. FedOT safeguards both the intelligent property of the model providers and the data privacy of the clients, while leveraging the distributed data for adapting LLMs to specific domains. This algorithm can be further integrated with the PEFT algorithms mentioned in Sec. 4.1.

4.3 Extendability by Unified Interfaces for Federated Fine-tuning

Notice that the aforementioned federated fine-tuning algorithms, regardless of whether the clients have access to the full model, are all implemented via a set of unified interfaces in FS-LLM. The interfaces compose a skeleton structure that can be customized to various FL applications. The interfaces can be invoked with customized functions in different stages, handling the underlying communication and synchronization between the server and clients in FL. The unified interfaces include but are not limited to the model pre-processing interface, initial model broadcast interface, shared parameter aggregation interface, and parameter re-distribution interface. We accommodate both closed-source and open-source

LLMs through a unified interface, built on FS event-driven architecture that simplifies the extension and customization of federated fine-tuning algorithms. Due to the space limit, we defer the detailed discussions about unified interfaces in Appendix C

5 LLM-Trainer: Training Operators and Paradigm

Although PEFT algorithms can significantly reduce the computation cost, they may still be computationally expensive for some clients with limited resources. To alleviate such concerns, we provide LLM-Trainer, which is designed to further accelerate computation and save resources during the local training and message transmission stages in FL. We implement a set of accelerating operators and resource-efficient operators to fulfill gap **(ii)**. Moreover, to meet different hardware settings or research goals, FS-LLM supports simulated mode, distributed mode, and clustered mode. For simulated mode, all clients run on a single machine to simulate the federated fine-tuning process. For the distributed mode and clustered mode, each client runs on one or more machines and communicates with the server separately. These modes share a consistent programming paradigm and behavior. Meanwhile, we note that there are also advanced FL problems in federated fine-tuning LLMs, such as pFL [58] and FedHPO [62]. Thus, to further fulfill gap **(iv)**, we implement LLM-Trainer as a collection of hook-like functions to support rich extensions, following the design principles of FS Trainer [65]. These hook-like functions will be executed by some arranged pattern to fine-tune the adapter within the local client. By adding, removing, or replacing hook-like functions, entities can conveniently customize local fine-tuning procedures and extend applicabilities. LLM-Trainer can be effortlessly compatible with implemented advanced FL algorithms for interdisciplinary research.

5.1 Training Operators for Acceleration and Efficiency

We introduce various accelerating operators and resource-efficient operators for fine-tuning LLMs in FS-LLM. These provided operators aim to optimize the federated fine-tuning process in terms of CPU/GPU memory consumption, multi-GPU parallel, and communication cost. We describe the operators and show how they can be combined to achieve better compatibility and efficiency.

Mode-generic operators. We provide operators generalized to different modes in the local fine-tuning stage of FS-LLM. We implement mixed-precision training and gradient accumulation operators in several hook-like functions to save GPU resources. Furthermore, to accelerate the local fine-tuning process and enable multi-GPU parallelism, FS-LLM integrates Pytorch's data parallel mechanism.

Mode-specific operators. Besides the aforementioned generalized operators, we develop specialized operators that are tailored for each mode and aim to address the bottlenecks in each mode. To be more specific, in the simulated mode, instantiating multiple clients on a single machine with several independent instantiated models causes a lot of memory consumption. Therefore, we use a round-robin switching operator, which allows the clients to take turns using the frozen full model for fine-tuning the adapter and then aggregating the updated adapters. Under this operator, when the number of clients grows, the memory consumption will only increase by an additional amount of the adapter. This improvement makes it possible to conduct simulated FL experiments with a number of clients on one single machine. For the distributed mode and clustered mode, we accelerate the communication between the server and the clients by one to two orders of magnitude by reducing the size of communication messages. We apply different communication optimizations for different messages and introduce communication-efficient operators, including quantization operator, streaming operator, and compression operator. Specifically, the quantization operator reduces the bit-width of the model parameters in the message to 16 or 8 bits; the streaming operator serializes the model parameters to eliminate the overhead of type conversion; the compression operator applies DEFLATE [15] or Gzip [16] algorithms to compress the messages.

Parallelization operators. Meanwhile, we also migrate the functionality of DeepSpeed as shown in Figure 2, providing data parallelism with multi-GPU memory optimization and CPU offloading capabilities, further enhancing resource utilization. Specifically, after launching the fine-tuning with DeepSpeed, we disable some modules (e.g., the logging module, the WandB module, and the file writing module) for the subprocess other than the Rank 0 process to avoid conflicts. In addition, in distributed and clustered modes, each subprocess communicates independently with each other across the server and the clients, and only synchronizes with other subprocesses during the local fine-tuning process, ensuring consistent behavior across different modes.

5.2 Training Paradigm for Extendability towards Advanced FL Tasks

Another design philosophy of LLM-Trainer is to support various extensions easily and integrate different hook-like functions with existing training paradigms seamlessly for FL interdisciplinary study. A vanilla fine-tuning process involves three steps: (1) preparing datasets and extracting them by batches, (2) iterating over the training datasets to update the model parameters, and (3) evaluating the performance of the fine-tuned model on validation datasets. We implement fine-grained learning behaviors for advanced FL tasks at different points in the above steps with hook-like functions. For instance, pFL [58] and FedHPO [62] are two advanced FL tasks that can significantly improve model performance in FL.

Adaptation of pFL with LLMs. FS [65] provides many implemented pFL plugins [8], which can be integrated for federated fine-finetuning LLMs. However, due to limited resources, it is unrealistic for the clients to maintain both global and local models at the same time for some pFL algorithms [33, 57], as it would consume a lot of memory. Therefore, we optimize the implementation of these pFL algorithms by only using global and local adapters. With such an implementation, by adding a pFL hook-like function, we can achieve the cooperation of any PEFT algorithms with the implemented pFL algorithms, which provides strong extensibility for federated fine-tuning LLMs in personalized settings.

Adaptation of FedHPO with LLMs. Similarly, for FedHPO, we offer extensions for FS-LLM with model-free HPO methods (e.g., random search [3] and grid search), model-based HPO methods (e.g., Bayesian Optimization [56]), multi-fidelity HPO method (e.g.,

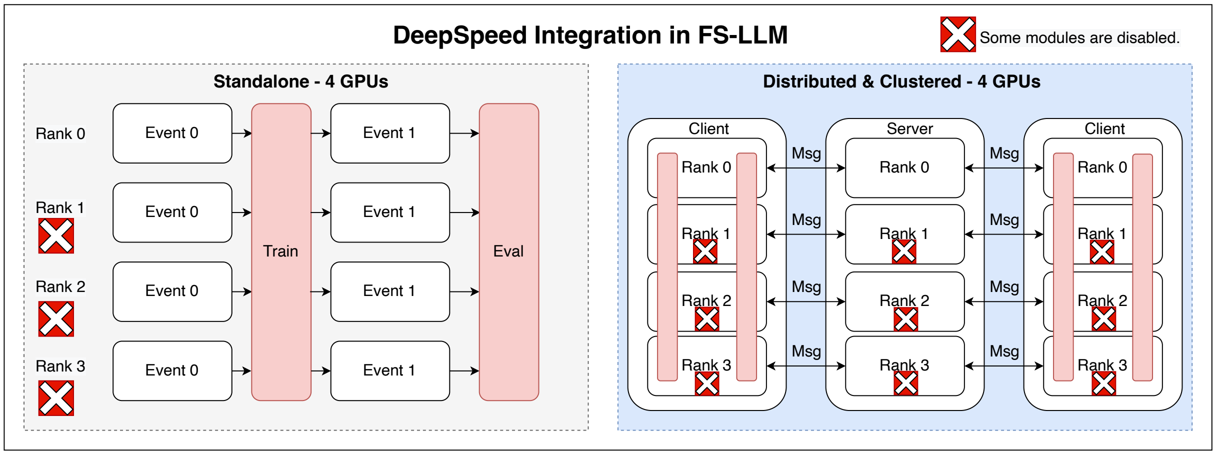

Figure 2: FS-LLM integrates DeepSpeed for federated fine-tuning in different hardware conditions. Rank 0 indicates the main process for multi-GPU training, and some modules of other subprocesses are disabled (e.g., logging and saving checkpoints). Msg stands for messages transmitted between the server and the clients, which trigger the events to happen.

Successive Halving Algorithm [25]), and FedHPO methods (e.g., FTS [14] and FLoRA [71]).

As our vision, the extensibility of FS-LLM to support the interdisciplinary study of LLMs and other FL scenarios dramatically expands the application scenarios of fine-tuning LLMs in FL and raises many new research opportunities, which we will discuss more details later in Sec. 7.

6 Experiments

In this section, we demonstrate the effectiveness of FS-LLM by comprehensive experiments with different algorithms and tasks. We want to answer the following research questions: (1) How effective and efficient is it to federated fine-tuning LLMs with PEFT algorithms (Sec. 6.1 and Sec. 6.2)? (2) How effective is it to federated fine-tune LLMs without accessing the full model (Sec. 6.3)? (3) What insights can we obtain from the interdisciplinary capabilities of FS-LLM in resolving pFL and FedHPO problems when federated fine-tuning LLMs (Sec. 6.4 and Appendix E)? We go through experimental results and provide answers to these questions.

6.1 Effectiveness of PEFT Algorithms in FS-LLM

We benchmark the performance of different PEFT algorithms in different application domains and scenarios. As described in Sec. 3, we use three federated fine-tuning datasets to fine-tune LLMs and evaluate them with corresponding tasks: (i) federated fine-tuning with *Fed-CodeAlpaca* for code generation and evaluating with *HumanEval*, (ii) federated fine-tuning with *Fed-Dolly* for generic language capability and evaluating with *HELM*, and (iii) federated fine-tuning with *Fed-GSM8K-3* for mathematical reasoning and evaluating with *GSM8K-test*. We conduct experiments in three scenarios: global (centralized fine-tuning), fed (federated fine-tuning), and local (separated fine-tuning). Specifically, the global scenario can be regarded as fine-tuning LLMs with one client who holds the whole fine-tuning dataset. Fed scenario means that clients federated fine-tune LLMs where each client holds a different fine-tuning dataset. Local scenario means that each client independently fine-tunes LLMs with its own fine-tuning dataset. All the experiments are conducted on the machines with the same hardware configuration three times with different random seeds. We report the average evaluation score with its standard deviation.

Benchmark federated fine-tuned LLaMA-7B. We use a widely adopted LLM, LLaMA-7B, with three PEFT algorithms (we exclude prefix-tuning from our experiments, because its implementation in [40] contains unresolved issues when we were preparing this package), including LoRA [22], P-tuning [38], and prompt tuning [32]. We employ FedAvg [41] as the federated aggregation strategy. To conduct the experiments uniformly and fairly, we fix the FL-specific hyperparameters and the hyperparameters that have a large impact on the computation cost for all experiments. Moreover, to further reduce the GPU memory consumption for efficiency, we employ the half-precision operator during fine-tuning. Please refer to Appendix D for experimental settings.

Results and Analysis. Table 1 shows the comparisons among different PEFT algorithms for federated fine-tuned LLaMA-7B under different scenarios. In summary, we can draw the following conclusions. (1) All algorithms with federated fine-tuning can significantly outperform those under the local scenario, and they all show very competitive results against those under the global scenario. This suggests that it is feasible and effective to federated fine-tuning LLMs with PEFT algorithms, which allows multiple entities to benefit from the collaborative training without directly sharing their private data. (2) Among these PEFT algorithms, LoRA shows the most promising performance and beats the other two algorithms by a large margin in all three scenarios. P-tuning and prompt tuning are two parameterized prompt algorithms that insert some learnable tokens into the model input to improve the performance of downstream tasks. However, they are outperformed by LoRA, a low-rank adaptation algorithm that augments each layer of LLMs with two low-rank matrices to capture new domain-specific knowledge. We conjecture that these parameterized prompt algorithms are limited by the knowledge encoded in the original LLM

Table 1: Perf. comparisons among different algorithms fine-tuning LLaMA-7B in FL: Evaluation Scores(%) ± standard deviation(%).

Algorithm	Scenario	*Fed-CodeAlpaca*	*Fed-Dolly*	*Fed-GSM8K-3*
LoRA	Global	13.54±0.24	46.25±0.44	14.81±1.04
	Fed	13.29±0.10	46.57±0.24	14.25±1.37
	Local	10.99±0.77	43.98±1.38	11.88±1.35
P-tuning	Global	10.24±0.30	41.29±0.01	12.13±0.41
	Fed	9.71±0.66	41.50±0.32	11.75±0.39
	Local	7.78±2.27	38.76±2.39	11.42±0.96
Prompt tuning	Global	9.80±1.79	41.24±0.54	9.75±1.49
	Fed	9.63±0.36	40.72±0.64	9.86±0.59
	Local	7.18±2.17	37.65±6.12	9.65±0.77

during the pre-training stage. This indicates that LoRA is suited for federate fine-tuning LLMs in future research or realistic scenarios.

How about federated fine-tuning previous-generation language models? As a comparison, we select the superior PEFT algorithm in Table 1, LoRA [22], to federated fine-tune a previous-generation language model, OPT-2.7B [68], which has fewer parameters than LLaMA-7B. Table 2 shows the evaluation results of OPT-2.7B on the *HumanEval*, *HELM* evaluations, and *GSM8K-test*.

Though the model performance of OPT-2.7B is improved with federated fine-tuning compared to those under the local scenario, it is not significant. Moreover, OPT-2.7B fails in the evaluation of some subtasks in HELM due to the exceeded input length, which limits the scope of the application of the previous-generation language models with fewer parameters. Thus, if the communication and computation costs are affordable, one should consider the larger scale of current-generation LLMs for more significant FL improvement and better absolute performance on downstream tasks. And please refer to the full version of FS-LLM [29] for more evaluation results of other LLMs, such as LLaMA-2-7B and more details about the performance comparision between LLaMA-7B and OPT-2.7B.

6.2 Efficiency of PEFT Algorithms in FS-LLM

In this section, we evaluate the efficiency of various PEFT algorithms in FL. Among all the metrics, we focus on their GPU memory consumption, message size, and computation time in the federated fine-tuning process. We note that the GPU memory consumption refers to the model only, excluding the input tokens and the optimizer state, because the input length varies greatly across data and affects the GPU memory consumption during fine-tuning LLMs. For message size, we report the number of bytes of the serialized adapter parameters during one communication between the server and one client. The computation time is defined as the time duration of one training step with a batch size of 1, from the start of forward propagation to the end of backward propagation. For a fair comparison, we follow the same experimental settings as Sec. 6.1 for all PEFT algorithms. All the experiments are conducted on two types of hardware, both with 512GB of RAM: Nvidia A100 GPU (80GB) with Intel Xeon Platinum 8369B CPU and Nvidia V100 GPU (32GB) with Intel Xeon Platinum 8163 CPU.

Results and Analysis. As shown in Table 3, we can first notice that (1) fine-tuning with different PEFT algorithms has negligible impact on the GPU memory consumption. (2) However, there are large differences in the message sizes, which in turn lead to large variations in the transmission time. For example, for a client with 100Mbps bandwidth, the transmission time (upload and download) per round ranges from about 0.01 seconds (with prompt tuning) to 40 seconds (with P-tuning) when using different PEFT algorithms. (3) Moreover, from the table, we can observe that the computation time varies due to different GPUs, with almost a twofold difference. Therefore, a more critical issue deserves attention: federated fine-tuning LLMs may suffer from more idle time due to the heterogeneity of computing resources among different clients. Because this computation efficiency difference can be significant, the benefit of mitigating communication latency by sending messages asynchronously in multiple computational stages may diminish.

Therefore, there are two research directions for federated fine-tuning LLMs in efficiency: (1) how to leverage the idle time of computation-resource-rich clients while they wait for computation-resource-limited clients to complete local updates, and (2) how to optimize the utilization of the available bandwidth resources in computation-resource-limited clients during computation.

6.3 Federated Fine-tuning LLMs without Accessing Full Model

In this section, we investigate the performance of federated fine-tuning LLMs without accessing the full model. As mentioned in Sec. 4.2, we adapt a privacy-preserving fine-tuning algorithm, offsite-tuning [64], to federated scenarios. Specifically, we use the first and last two layers of LLaMA-7B as the adapter and compress the model as the emulator by dropping 20% and 50% of the remaining layers uniformly. Then the server broadcasts both the adapter and emulator to all clients, and the clients only fine-tune the adapter with FedAvg. We compare the performance of LLMs with fine-tuned adapters via federated offsite-tuning (denoted as FedOT) and corresponding local offsite-tuning (denoted as LocalOT). Following Sec. 6.1, we benchmark FedOT on LLM-Benchmarks.

Results and Analysis. We present the evaluation scores of FedOT and LocalOT in Table 4. We can have the following observations: (1) Comparing FedOT and LocalOT, FL offers significant benefits for this privacy-preserving fine-tuning algorithm for federated fine-tuning LLMs without accessing the full model. This demonstrates that FedOT can still enable multiple entities to benefit from collaborative training without sharing their private data directly when they cannot access the full model. (2) When the dropping rate is

Table 2: Performance of federated fine-tuned previous-generation language model, OPT-2.7B, with LoRA: Evaluation Scores(%) ± standard deviation(%). (*OPT-2.7B fails on some subtasks in HELM due to the exceeded input length, and failed subtasks are excluded when calculating the final results.)

Algorithm	Scenario	*Fed-CodeAlpaca*	*Fed-Dolly*	*Fed-GSM8K-3*
LoRA	Global	0.61±0.61	25.90±0.40*	2.92±0.11
	Fed	0.43±0.07	25.90±0.33*	2.88±0.17
	Local	0.25±0.25	25.06±1.02*	2.25±0.22

Table 3: Efficiency comparisons among different PEFT algorithms when fine-tuning LLaMA-7B in FL. (*The GPU usage is subject to minor variations due to the differences in Cuda versions.)

	Full	LoRA	P-tuning	Prompt tuning
GPU Usage* (MB)	13,440	13,450	13,538	13,442
Message Size (MB)	12,852.01	21.40	256.48	0.17
Comp. Time on A100 (Sec.)	0.19±0.01	0.16±0.02	0.15±0.03	0.15±0.04
Comp. Time on 3090 (Sec.)	OOM	0.18±0.05	0.18±0.04	0.16±0.05
Comp. Time on V100 (Sec.)	OOM	0.33±0.07	0.33±0.08	0.33±0.10

20%, FedOT still achieves competitive performance compared to some of those PEFT algorithms, even though the clients cannot access the full model. However, it should be noted that this is because the number of parameters of the adapter in FedOT is significantly larger than those in the PEFT algorithms. FedOT sacrifices communication efficiency for model performance, making it effective in scenarios where clients cannot access the full model. (3) On the other hand, the results show that when the dropping rate increases to 50% from 20%, the model loses a large amount of knowledge from the pre-training stage, almost fails to retain the capacity of the CoT and code generation, and hardly acquire new knowledge from fine-tuning. There is a trade-off between the compression rate and the performance of LLMs: increasing the compression rate enhances the privacy of LLMs but degrades their performance. While an increase in compression rate can lead to a decline in model performance, this compression serves a dual purpose. On the one hand, it ensures efficiency in the FL, and on the other, it builds a more substantial barrier against potential reconstruction of model by downstream clients. This correlation acts as a mechanism to protect model's privacy, preventing disclosure of LLM's parameters. This indicates that compressing LLMs while maintaining their generalization ability and privacy is a promising direction to explore.

6.4 Personalized Federated Learning for LLMs

To explore the potential of personalized federated learning algorithms [58] for fine-tuning LLMs in FL, we compare pFedMe [57] and FedAvg [41] with LoRA in this subsection under different data heterogeneity. To simulate different levels of data heterogeneity, we create variants of *Fed-CodeAlpaca*.

We use the programming language of the code samples as the label and split the fine-tuning dataset into nine clients by Latent Dirichlet Allocation (i.e., Dirichlet splitter). We use four different values for α of the Dirichlet splitter from 0.05 to 50.0. Along with the original *Fed-CodeAlpaca* dataset, we obtain five federated fine-tuning datasets with different heterogeneity. Following Sec. 6.1, we repeat each experiment three times with different random seeds. The averaged evaluation scores (Pass@1 scores) with their standard deviation are reported. We note that evaluation scores with pFedMe are obtained by benchmarking each personalized client individually and then computing their average scores.

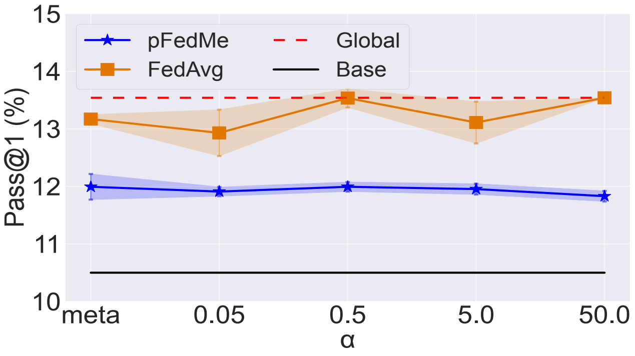

Figure 3: The comparison of performance with pFedMe and FedAvg over different data heterogeneity: the higher α, the lower the data heterogeneity among clients. Global stands for fine-tuning LLMs under the global scenario, and Base stands for the original model, which is not fine-tuned.

Results and Analysis. We present the performance of fine-tuning with LoRA using FedAvg and pFedMe in different data heterogeneities in Figure 3, which shows the performance of fine-tuning with FedAvg gradually approaches that under the global scenario as data heterogeneity decreases. However, pFedMe surprisingly does not outperform FedAvg under any data distribution, which shows different results with previous language models in pFL-bench [8]. We analyze this phenomenon and find that: (1) To improve efficiency, we use the half-precision operator to fine-tune LLMs in the experiment. However, this leads to a more pronounced precision

Table 4: Performance comparisons between different compression rates (dropping layers uniformly) when fine-tuning LLaMA-7B without accessing the full model under federated and local scenarios: Evaluation Scores(%) ± standard deviation(%).

Dropping Rate	Scenario	*Fed-CodeAlpaca*	*Fed-Dolly*	*Fed-GSM8K-3*
20%	Fed	7.14±2.75	44.88±0.75	9.02±0.71
	Local	0.18±0.50	38.45±9.57	4.72±2.91
50%	Fed	0.16±0.15	37.01±2.34	2.98±0.98
	Local	0.00±0.00	35.44±5.99	1.82±1.29

loss for pFedMe than FedAvg, since pFedMe multiplies the update of the local model with respect to the global model by a small learning rate. This adversely impacts the performance of pFedMe. (2) The use of acceleration operators for efficient LLM fine-tuning restricts the range of hyperparameter space in these pFL algorithms, affecting the upper bound of algorithm performance in the valid parameter space. For example, assuming the LLM performs better when the learning rate is 10^{-5}, but at this point, using half-precision or mixed-precision training for efficiency, the model can barely be updated due to the precision loss.

Based on these observations, we believe how to ensure the compatibility of various efficient training operators and different pFL algorithms is still unclear and deserves more attention from the community. Besides, sharing a common base LLM and only maintaining multiple versions of adapters may not be compatible with some existing pFL algorithms because pFL algorithms may introduce access conflicts. For instance, when a penalty term is used to constrain the large updates of the local model, the computation to compute updates requires using both the global adapter with the base LLM and the local adapter with the base LLM in the same step. Significantly, more memory cost is required to avoid the conflict by maintaining multiple copies of LLMs and their adapters. Resolving this challenge may require new algorithm development or a new pFL computation pattern for future work.

7 Conclusions

In this paper, we identify gaps that need to be addressed between fine-tuning LLMs in federated settings and the existing universal FL frameworks. To bridge these gaps, we introduce our open-source package, FS-LLM, with rich functionalities and extensibilities, which supports federated fine-tuning LLMs under various FL scenarios. We conduct extensive experiments to demonstrate the utility of our package and gain insights into how to fine-tune LLMs in constrained environments (FL settings).

Based on our observations, we outline promising directions for future research in federated fine-tuning LLMs:

- Designing computation-efficient fine-tuning algorithms for federated fine-tuning LLMs. Even with PEFT algorithms, the computation cost is still too high for resource-limited clients. Reducing the computation cost can lower the barrier for data holders and allow more entities to benefit from the federated fine-tuning LLMs.
- Exploring more privacy-preserving fine-tuning algorithms without accessing the full model in FL. FedOT suffers from a trade-off between model compression rate and model performance. Addressing this issue would protect the sensitive information of LLMs in FL from exposing pre-training data and the valuable full model, which could be exploited by malicious entities for adversarial attacks or commercial gains.
- Optimizing pFL algorithms to enable robust combination with accelerating and resource-efficient operators. If performance degradation due to low-precision training can be overcome, it would improve the personalized model performance when data are heterogeneous and computation resources are limited.
- Investigating low-fidelity FedHPO methods for fine-tuning LLMs in FL. Based on our experimental results, we find the inconsistency between validation loss and the generalization performance of LLMs. Overcoming this inconsistency would help find optimal hyperparameters for fed fine-tuning LLMs with low cost, resulting in better generalization performance.
- Extending the federated LLM fine-tuning to cross-device setting. As we have already observed the demand in the cross-silo scenario, we also notice a similar need in the cross-device scenario [9, 20, 30]. In the cross-device scenario, the clients are more numerous and heterogeneous, the computational resources are more limited, and the network conditions are more diverse. How to federated fine-tune LLMs under the cross-device scenario is an urgent problem that needs to be solved.

For promoting further research and deploying federated fine-tuning LLMs in constrained environments, we release FS-LLM at https://github.com/alibaba/FederatedScope/tree/llm.

References

[1] Anthropic. 2023. Introducing Claude.

[2] Jiamu Bai, Daoyuan Chen, Bingchen Qian, Liuyi Yao, and Yaliang Li. 2024. Federated Fine-tuning of Large Language Models under Heterogeneous Language Tasks and Client Resources. *arXiv preprint arXiv:2402.11505* (2024).

[3] James Bergstra and Yoshua Bengio. 2012. Random Search for Hyper-Parameter Optimization. *Journal of machine learning research* 13 (2012), 281–305.

[4] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, et al. 2019. Towards federated learning at scale: System design. In *Proc. of machine learning and systems (MLSys'19)*, Vol. 1. 374–388.

[5] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language Models are Few-Shot Learners. In *Proc. of the Advances in Neural Information Processing Systems (NeurIPS'20)*, Vol. 33. 1877–1901.

[6] Sebastian Caldas, Sai Meher Karthik Duddu, Peter Wu, Tian Li, Jakub Konečnỳ, H Brendan McMahan, Virginia Smith, and Ameet Talwalkar. 2018. Leaf: A benchmark for federated settings. *arXiv preprint arXiv:1812.01097* (2018).

[7] Sahil Chaudhary. 2023. Code Alpaca: An Instruction-following LLaMA model for code generation. https://github.com/sahil280114/codealpaca.

[8] Daoyuan Chen, Dawei Gao, Weirui Kuang, Yaliang Li, and Bolin Ding. 2022. pFL-Bench: A Comprehensive Benchmark for Personalized Federated Learning. In *Proc. of the Advances in Neural Information Processing Systems (NeurIPS'22)*, Vol. 35. 9344–9360.

[9] Daoyuan Chen, Dawei Gao, Yuexiang Xie, Xuchen Pan, Zitao Li, Yaliang Li, Bolin Ding, and Jingren Zhou. 2023. FS-Real: Towards Real-World Cross-Device Federated Learning. In *Proc. of the ACM SIGKDD International Conference on*

Knowledge Discovery and Data Mining (KDD'23). 3829–3841.

[10] Mark Chen, Jerry Tworek, Heewoo Jun, et al. 2021. Evaluating Large Language Models Trained on Code. *arXiv perprint arXiv:2107.03374* (2021).

[11] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, et al. 2022. PaLM: Scaling Language Modeling with Pathways. *arXiv preprint arXiv:2204.02311* (2022).

[12] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, et al. 2021. Training Verifiers to Solve Math Word Problems. *arXiv preprint arXiv:2110.14168* (2021).

[13] Mike Conover, Matt Hayes, Ankit Mathur, Jianwei Xie, Jun Wan, Sam Shah, Ali Ghodsi, Patrick Wendell, Matei Zaharia, and Reynold Xin. 2023. Free Dolly: Introducing the World's First Truly Open Instruction-Tuned LLM.

[14] Zhongxiang Dai, Bryan Kian Hsiang Low, and Patrick Jaillet. 2020. Federated Bayesian Optimization via Thompson Sampling. In *Proc. of the Advances in Neural Information Processing Systems (NeurIPS'20)*, Vol. 33. 9687–9699.

[15] L. Peter Deutsch. 1996. *DEFLATE Compressed Data Format Specification version 1.3.* RFC. Network Working Group.

[16] L. Peter Deutsch. 1996. *GZIP file format specification version 4.3.* RFC. Network Working Group.

[17] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proc. of NAACL-HLT (NAACL-HLT'19)*. 4171–4186.

[18] Chenhe Dong, Yuexiang Xie, Bolin Ding, Ying Shen, and Yaliang Li. 2022. Collaborating Heterogeneous Natural Language Processing Tasks via Federated Learning. *arXiv preprint arXiv:2212.05789* (2022).

[19] Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, et al. 2023. PaLM-E: An Embodied Multimodal Language Model. *arXiv preprint arXiv:2303.03378* (2023).

[20] Dawei Gao, Daoyuan Chen, Zitao Li, Yuexiang Xie, Xuchen Pan, Yaliang Li, Bolin Ding, and Jingren Zhou. 2023. FS-Real: A Real-World Cross-Device Federated Learning Platform. *PVLDB* 16 (2023), 4046–4049.

[21] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-Efficient Transfer Learning for NLP. In *Proc. of the International Conference on Machine Learning (ICML'21)*. 2790–2799.

[22] Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. 2022. LoRA: Low-Rank Adaptation of Large Language Models. In *Proc. of the International Conference on Learning Representations (ICLR'22)*.

[23] Shaohan Huang, Li Dong, Wenhui Wang, Yaru Hao, Saksham Singhal, Shuming Ma, Tengchao Lv, Lei Cui, Owais Khan Mohammed, Qiang Liu, et al. 2023. Language Is Not All You Need: Aligning Perception with Language Models. *arXiv preprint arXiv:2302.14045* (2023).

[24] Daphne Ippolito, Ann Yuan, Andy Coenen, and Sehmon Burnam. 2022. Creative Writing with an AI-Powered Writing Assistant: Perspectives from Professional Writers. *arXiv preprint arXiv:2211.05030* (2022).

[25] Kevin Jamieson and Ameet Talwalkar. 2016. Non-stochastic Best Arm Identification and Hyperparameter Optimization. In *Proc. of the Artificial intelligence and statistics (AISTATS'16)*. 240–248.

[26] Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of Hallucination in Natural Language Generation. *Comput. Surveys* 55 (2023), 1–38.

[27] Rabeeh Karimi Mahabadi, James Henderson, and Sebastian Ruder. 2021. Compacter: Efficient Low-Rank Hypercomplex Adapter Layers. In *Proc. of the Advances in Neural Information Processing Systems (NeurIPS'21)*, Vol. 34. 1022–1035.

[28] Jakub Konečný, H. Brendan McMahan, Daniel Ramage, and Peter Richtárik. 2016. Federated Optimization: Distributed Machine Learning for On-Device Intelligence. *arXiv preprint arXiv:1610.02527* (2016).

[29] Weirui Kuang, Bingchen Qian, Zitao Li, Daoyuan Chen, Dawei Gao, Xuchen Pan, Yuexiang Xie, Yaliang Li, Bolin Ding, and Jingren Zhou. 2023. FederatedScope-LLM: A Comprehensive Package for Fine-tuning Large Language Models in Federated Learning. arXiv:2309.00363

[30] Fan Lai, Yinwei Dai, Sanjay Singapuram, Jiachen Liu, Xiangfeng Zhu, Harsha Madhyastha, and Mosharaf Chowdhury. 2022. FedScale: Benchmarking Model and System Performance of Federated Learning at Scale. In *Proc. of the International Conference on Machine Learning (ICML'22)*, Vol. 162. 11814–11827.

[31] Mina Lee, Percy Liang, and Qian Yang. 2022. CoAuthor: Designing a Human-AI Collaborative Writing Dataset for Exploring Language Model Capabilities. In *Proc. of the CHI Conference on Human Factors in Computing Systems (CHI'22)*. 19 pages.

[32] Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The Power of Scale for Parameter-Efficient Prompt Tuning. In *Proc. of the Conference on Empirical Methods in Natural Language Processing (EMNLP'21)*. 3045–3059.

[33] Tian Li, Shengyuan Hu, Ahmad Beirami, and Virginia Smith. 2021. Ditto: Fair and Robust Federated Learning Through Personalization. In *Proc. of the International Conference on Machine Learning (ICML'21)*, Vol. 139. 6357–6368.

[34] Xiang Lisa Li and Percy Liang. 2021. Prefix-Tuning: Optimizing Continuous Prompts for Generation. In *Proc. of the Annual Meeting of the Association for Computational Linguistics and the International Joint Conference on Natural Language Processing (ACL-IJCNLP'21)*. 4582–4597.

[35] Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, et al. 2022. Holistic Evaluation of Language Models. *arXiv preprint arXiv:2211.09110* (2022).

[36] Zhenqing Ling, Daoyuan Chen, Liuyi Yao, Yaliang Li, and Ying Shen. 2024. On the Convergence of Zeroth-Order Federated Tuning for Large Language Models. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*.

[37] Xiao Liu, Kaixuan Ji, Yicheng Fu, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2021. P-Tuning v2: Prompt Tuning Can Be Comparable to Fine-tuning Universally Across Scales and Tasks. *arXiv preprint arXiv:2110.07602* (2021).

[38] Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2021. GPT Understands, Too. *arXiv preprint arXiv:2103.10385* (2021).

[39] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv preprint arXiv:1907.11692* (2019).

[40] Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, and Sayak Paul. 2022. PEFT: State-of-the-art Parameter-Efficient Fine-Tuning methods. https://github.com/huggingface/peft.

[41] H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *Proc. of the Artificial intelligence and statistics (AISTATS'17)*. 1273–1282.

[42] Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, et al. 2018. Mixed Precision Training. In *Proc. of the International Conference on Learning Representations (ICLR'18)*.

[43] OpenAI. 2022. Introducing ChatGPT.

[44] OpenAI. 2023. GPT-4 Technical Report. *arXiv preprint arXiv:2303.08774* (2023).

[45] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Proc. of the Advances in Neural Information Processing Systems (NeurIPS'19)*, Vol. 32. 8024–8035.

[46] Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. 2023. Gorilla: Large Language Model Connected with Massive APIs. *arXiv preprint arXiv:2305.15334* (2023).

[47] Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. 2020. AdapterFusion: Non-Destructive Task Composition for Transfer Learning. *arXiv preprint arXiv:2005.00247* (2020).

[48] Jonas Pfeiffer, Ivan Vulić, Iryna Gurevych, and Sebastian Ruder. 2020. MAD-X: An Adapter-Based Framework for Multi-Task Cross-Lingual Transfer. In *Proc. of the Conference on Empirical Methods in Natural Language Processing (EMNLP'20)*. 7654–7673.

[49] Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, et al. 2023. ToolLLM: Facilitating Large Language Models to Master 16000+ Real-world APIs. *arXiv preprint arXiv:2307.16789* (2023).

[50] Zhen Qin, Daoyuan Chen, Bingchen Qian, Bolin Ding, Yaliang Li, and Shuiguang Deng. 2024. Federated Full-Parameter Tuning of Billion-Sized Language Models with Communication Cost under 18 Kilobytes. In *International Conference on Machine Learning*.

[51] Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. 2020. DeepSpeed: System Optimizations Enable Training Deep Learning Models with Over 100 Billion Parameters. In *Proc. of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'20)*. 3505–3506.

[52] Jie Ren, Samyam Rajbhandari, Reza Yazdani Aminabadi, Olatunji Ruwase, Shuangyang Yang, Minjia Zhang, Dong Li, and Yuxiong He. 2021. ZeRO-Offload: Democratizing Billion-Scale Model Training. In *Proc. of the USENIX Annual Technical Conference (USENIX ATC'21)*. 551–564.

[53] Gene Ruebsamen. 2023. Cleaned Alpaca Dataset. https://github.com/gururise/AlpacaDataCleaned.

[54] Théo Ryffel, Andrew Trask, Morten Dahl, Bobby Wagner, Jason V. Mancuso, Daniel Rueckert, and Jonathan Passerat-Palmbach. 2018. A generic framework for privacy preserving deep learning. *arXiv preprint arXiv:1811.04017* (2018).

[55] Teven Le Scao, Angela Fan, Christopher Akiki, et al. 2022. BLOOM: A 176B-Parameter Open-Access Multilingual Language Model. *arXiv preprint arXiv:2211.05100* (2022).

[56] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando De Freitas. 2015. Taking the Human Out of the Loop: A Review of Bayesian Optimization. *Proc. IEEE* 104 (2015), 148–175.

[57] Canh T Dinh, Nguyen Tran, and Josh Nguyen. 2020. Personalized Federated Learning with Moreau Envelopes. In *Proc. of the Advances in Neural Information Processing Systems (NeurIPS'20)*, Vol. 33. 21394–21405.

[58] Alysa Ziying Tan, Han Yu, Lizhen Cui, and Qiang Yang. 2022. Towards Personalized Federated Learning. *IEEE Transactions on Neural Networks and Learning Systems* (2022), 1–17.

[59] Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford Alpaca: An Instruction-following LLaMA model. https://github.com/tatsu-lab/stanford_alpaca.

[60] Hugo Touvron, Thibaut Lavril, Gautier Izacard, et al. 2023. LLaMA: Open and Efficient Foundation Language Models. *arXiv preprint arXiv:2302.13971* (2023).

[61] Zhen Wang, Weirui Kuang, Yuexiang Xie, Liuyi Yao, Yaliang Li, Bolin Ding, and Jingren Zhou. 2022. FederatedScope-GNN: Towards a Unified, Comprehensive and Efficient Package for Federated Graph Learning. In *Proc. of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'22)*.

[62] Zhen Wang, Weirui Kuang, Ce Zhang, Bolin Ding, and Yaliang Li. 2023. FedHPO-Bench: A Benchmark Suite for Federated Hyperparameter Optimization. In *Proc. of the International Conference on Machine Learning (ICML'23)*. 35908–35948.

[63] Chenfei Wu, Sheng-Kai Yin, Weizhen Qi, Xiaodong Wang, Zecheng Tang, and Nan Duan. 2023. Visual ChatGPT: Talking, Drawing and Editing with Visual Foundation Models. *arXiv preprint arXiv:2303.04671* (2023).

[64] Guangxuan Xiao, Ji Lin, and Song Han. 2023. Offsite-Tuning: Transfer Learning without Full Model. *arXiv preprint arXiv:2302.04870* (2023).

[65] Yuexiang Xie, Zhen Wang, Daoyuan Chen, Dawei Gao, Liuyi Yao, Weirui Kuang, Yaliang Li, Bolin Ding, and Jingren Zhou. 2023. FederatedScope: A Flexible Federated Learning Platform for Heterogeneity. *PVLDB* 16 (2023), 1059–1072.

[66] Qiang Yang, Yang Liu, Yong Cheng, Yan Kang, Tianjian Chen, and Han Yu. 2019. Federated Learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning* 13 (2019), 1–207.

[67] Aohan Zeng, Xiao Liu, Zhengxiao Du, Zihan Wang, Hanyu Lai, Ming Ding, Zhuoyi Yang, Yifan Xu, Wendi Zheng, Xiao Xia, et al. 2023. GLM-130B: An Open Bilingual Pre-trained Model. In *Proc. of the International Conference on Learning Representations (ICLR'23)*.

[68] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. 2022. OPT: Open Pre-trained Transformer Language Models. *arXiv preprint arXiv:2205.01068* (2022).

[69] Xishan Zhang, Shaoli Liu, Rui Zhang, Chang Liu, Di Huang, Shiyi Zhou, Jiaming Guo, Yu Kang, Qi Guo, Zidong Du, and Yunji Chen. 2019. Adaptive Precision Training: Quantify Back Propagation in Neural Networks with Fixed-point Numbers. *arXiv preprint arXiv:1911.00361* (2019).

[70] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. 2023. A Survey of Large Language Models. *arXiv preprint arXiv:2303.18223* (2023).

[71] Yi Zhou, Parikshit Ram, Theodoros Salonidis, Nathalie Baracaldo, Horst Samulowitz, and Heiko Ludwig. 2023. Single-shot Hyper-parameter Optimization for Federated Learning. In *Proc. of the International Conference on Learning Representations (ICLR'23)*.

Appendix

A The Scope of FS-LLM

The landscape of FL is rapidly evolving, especially with the increasing attention on leveraging LLMs to harness decentralized data while preserving privacy. However, the complexity and heterogeneity of FL scenarios pose unique challenges, impeding the efficient utilization and robust evaluation of LLMs across diverse applications. This paper endeavors to bridge the gap between the theoretical potential and practical utility of FL with LLMs. Our contributions address critical aspects of this endeavor, from fine-tuning methodologies to evaluation frameworks, and from computational efficiency to privacy considerations.

To consolidate the implementations and facilitate the FL research about LLMs, one of the critical steps is to construct a module with an end-to-end benchmarking pipeline, i.e., LLM-Benchmarks in FS-LLM. We assemble various corpus datasets from different domains for fine-tuning and pair each of them with one specific relevant evaluation task to assess the performance of the fine-tuned LLMs on different domains. For the datasets prepared for fine-tuning, we partition them according to their meta-information, formulating them into federated versions. Meanwhile, FS-LLM offers *Splitter* to split centralized datasets, enhancing the extensibility of the fine-tuning dataset for federated fine-tuning LLMs. Furthermore, we provide a rich set of docker images where the runtime environment of the evaluation tasks has been prepared. LLM-Benchmarks enables users to conveniently compare the effectiveness of different fine-tuning algorithms in various FL scenarios. Another important and helpful component provided in FS-LLM, LLM-AlgZoo, includes a collection of fine-tuning algorithms tailored for FL. As communication and computation resources of clients are usually limited in FL, we leverage and integrate several PEFT algorithms into the FL setting, such as LoRA [22], prefix-tuning [34], P-tuning [38], and prompt tuning [32]. Compared to full-parameter fine-tuning, these PEFT algorithms significantly reduce memory consumption, training time, and communication costs for fine-tuning LLMs. Besides, motivated by the practical concerns on intelligent property protection of LLMs, we also integrate a privacy-preserving fine-tuning algorithm, offsite-tuning [64], for the scenario where clients only tune small adapters based on a distilled model from a full LLM. Based on the fine-tuning algorithms mentioned above (i.e., the PEFT algorithms and offsite-tuning), LLM-Trainer powers the FL fine-tuning process by adopting a hook-like training scheme. Though these algorithms fine-tune a small number of parameters of LLMs, they can still be computationally expensive. Therefore, FS-LLM incorporates various accelerating operators and resource-efficient operators as hook-like functions, such as DeepSpeed's ZeRO [51], Pytorch's data parallelism [45], and model quantization [69], to accelerate the local fine-tuning process and enable FS-LLM to work efficiently on consumer-level GPUs. Besides, LLM-Trainer, with the help of its customizable hook-like programming interfaces, can be quickly and seamlessly integrated with existing plug-ins in FS for interdisciplinary studies, such as pFL [58] and FedHPO [62], which could be a promising direction for further research. For meeting different hardware settings or research goals, LLM-Trainer enables FS-LLM to fine-tune LLMs in simulated mode (single-machine FL for simulation, all clients in one machine), distributed mode (multi-machine FL, one client per machine), and clustered mode (multi-machine FL, one client per computation cluster).

B Dataset Description

In this section, we describe the fine-tuning datasets curated in FS-LLM, and summarize the statistics and information in Table 5. These datasets are derived from existing and widely used fine-tuning datasets that cover diverse domains, such as code, natural language, dialogues, and math problems. The curated fine-tuning datasets exhibit different degrees of heterogeneity across clients, which pose various challenges and opportunities for fine-tuning LLMs in FL. We describe the construction and characteristics of each dataset in detail below and illustrate their scenarios. We will constantly adopt new datasets for fine-tuning LLMs in FS-LLM.

Table 5: Statistics and information of the fine-tuning datasets.

Name	#Client	#Sample	Split	Domain	Evaluation Dataset
Fed-CodeAlpaca	9	~8.0k	Meta	Code Generation	*HumanEval*
Fed-Dolly	8	~15.0k	Meta	Generic Language	*HELM*
Fed-GSM8K-3	3	~7.5k	IID	CoT	*GSM8K-test*
Fed-CodeSearchNet	6	~1880.8k	Meta	Code Generation	*HumanEval*
Alpaca	-	~52.0k	-	Generic Language	*HELM*
CleanedAlpaca	-	~51.8k	-	Generic Language	*HELM*

C Discussion about the Unified Interfaces

Figure 4 illustrates unified interfaces used by fine-tuning algorithms in Sec. 4.1 and 4.2. The unified interfaces include but are not limited to the model pre-processing interface (arrow ①), initial model broadcast interface (arrow ②), shared parameter aggregation interface (arrow ③), and parameter re-distribution interface (arrow ④). For closed-source LLMs, which are inaccessible to clients, the model providers can compress the LLM into an emulator by implementing a distillation function with the model pre-processing interface; for open-source LLMs, which are accessible, the pre-processing interface returns the original LLM. These unified but flexible interfaces are based on the event-driven architecture in FL [65]. In the context of federated LLM fine-tuning, events are the exchanged information (e.g., local updated weights). Each event has a corresponding handler, which is the actions triggered by it. The event-driven architecture allows the entities of FL to program their behaviors and react to the events. Thus, by designing message-handler pairs, users can extend and customize the federated fine-tuning algorithms easily because of the extensibility of FS-LLM.

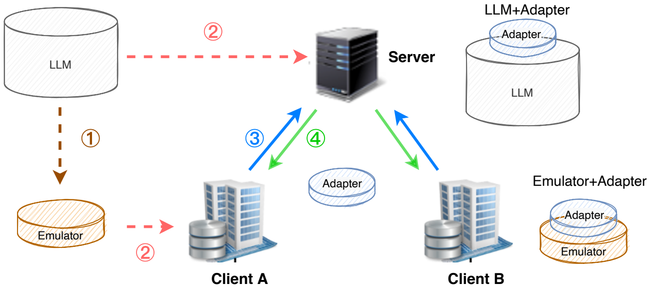

Figure 4: The unified interfaces for federated fine-tuning LLMs with or without accessing the full model.

D Experiment Protocols and Results

We provide experimental configurations in this section so that interested readers can reproduce our experimental results. All the experiments are conducted on the machines with the same hardware configuration: Nvidia A100 GPU (80GB) with Intel Xeon Platinum 8369B CPU and 512GB of RAM. For all scenarios, we repeat the experiments three times with different random seeds. We report the average evaluation score with its standard deviation. For all experiments, we set the communication round to 500, the local update step to 30, and the batch size to 1. We perform a grid search for algorithm-specific and learning-related hyperparameters to obtain the optimal configuration. For example, the search space of the learning rate is $\{1\times10^{-4}, 3\times10^{-4}, 5\times10^{-4}, 1\times10^{-3}, 3\times10^{-3}, 5\times10^{-3}\}$. Due to the space limit, please refer to the full version of FS-LLM paper [29] for detailed experimental results with PEFT algorithms under different hyperparameters in FL.

E Study of FedHPO for LLMs

Since fine-tuning LLMs in FL is very costly, it is usually infeasible to perform full-fidelity hyperparameter optimization. However, we observe that the performance of fine-tuned LLMs in FL is highly dependent on the choice of hyperparameters. Therefore, we investigate whether we can use low-fidelity FedHPO [62] methods in this scenario. We follow the experiment settings in Sec. 6.1 and use LoRA to fine-tune LLaMA-7B on *Fed-CodeAlpaca* yet with lower fidelity (i.e., fewer communication rounds). We rank all the hyperparameter combinations searched by their validation loss in ascending order and evaluation scores in descending order separately. We plot the two landscapes of the rank to explore the feasibility of using low-fidelity FedHPO methods when federated fine-tuning LLMs.

Results and Analysis. Based on the results shown in Figure 5, we distill the following observations. (1) We observe that the rank of the evaluation scores of the fine-tuned LLMs varies drastically and non-smoothly with respect to the hyperparameter changes. This poses a great challenge for finding the optimal hyperparameters, as it requires a fine-grained and exhaustive search over the hyperparameter space. (2) Moreover, we reveal a significant discrepancy between the ranks of validation loss and the ranks of final generalization performance in evaluation tasks during fine-tuning. This implies that the validation loss may not reflect the generalization ability of the fine-tuned LLMs.

In summary, we uncover two major challenges for fine-tuning LLMs in FedHPO: (1) the evaluation scores are highly sensitive and non-smooth to the hyperparameter changes, and (2) the validation loss may not be a reliable indicator of the generalization performance. These challenges identify two promising yet unexplored research directions for future work on fine-tuning LLMs in FedHPO. The first direction is to develop fine-grained but efficient FedHPO methods for finding the optimal hyperparameters on federated fine-tuning LLMs, which can avoid exhaustive searches over the hyperparameter space. The second direction is to exploit concurrent exploration in FedHPO to evaluate the generalization ability of the client-side hyperparameters with low fidelity for each client during the FL process.

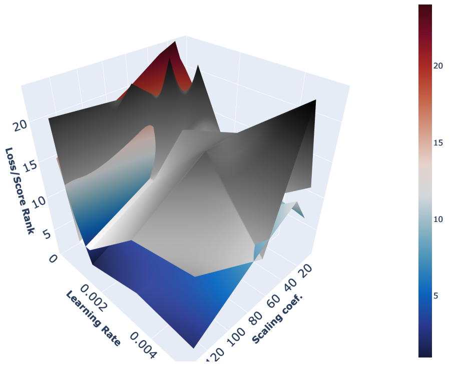

Figure 5: The landscape of the rank of the validation loss and the Pass@1 scores over all the hyperparameter combinations. The grey surface shows the distribution of the rank based on the Pass@1 scores, and the blue-red surface indicates the distribution of the rank based on the validation loss.