Journal of Machine Learning Research 24 (2023) 1-7 Submitted 4/28; Revised 10/22; Published 1/23

FedLab: A Flexible Federated Learning Framework

Dun Zeng[1,2,†] ZENGDUN@STD.UESTC.EDU.CN
Siqi Liang[4,†] ZSZXLSQ@GMAIL.COM
Xiangjing Hu[3] XIANGJINGHU@STU.HIT.EDU.CN
Hui Wang[2] WANGH06@PCL.AC.CN
Zenglin Xu[3,2,⋆] XUZENGLIN@HIT.EDU.CN

[1] *University of Electronic Science and Technology of China*
[2] *Peng Cheng Lab*
[3] *Harbin Institute of Technology Shenzhen*
[4] *Shenzhen Research Institute, The Chinese University of Hong Kong*
[†] *Denotes equal contribution*
[⋆] *Corresponding author*

Editor: Sebastian Schelter

Abstract

`FedLab` is a lightweight open-source framework for the simulation of federated learning. The design of `FedLab` focuses on federated learning algorithm effectiveness and communication efficiency. It allows customization on server optimization, client optimization, communication agreement, and communication compression. Also, `FedLab` is scalable in different deployment scenarios with different computation and communication resources. We hope `FedLab` could provide flexible APIs as well as reliable baseline implementations and relieve the burden of implementing novel approaches for researchers in the FL community. The source code, tutorial, and documentation can be found at `https://github.com/SMILELab-FL/FedLab`.

Keywords: distributed learning, federated learning, Python, PyTorch

1. Introduction

Federated learning (FL) is recently a burgeoning research area of machine learning (McMahan et al., 2017). It aims to protect individual data privacy in distributed machine learning applications, especially in finance (Byrd and Polychroniadou, 2020), smart healthcare, (Xu et al., 2021; Brisimi et al., 2018) and edge computing (Jiang et al., 2019; Meng et al., 2021). Unlike traditional data-centered distributed machine learning, participants in the FL settings utilize localized data to train local models and then leverage aggregation strategies with other participants to collaboratively acquire a global model without direct data-sharing.

The training scheme of most FL methods is generally composed of four steps in each round: 1. The server selects a subset of clients and broadcasts global information (e.g., global model parameters). 2. Each client updates the local model with a private dataset. 3. Clients upload local information (e.g., model parameters) to the server. 4. The server updates the global model based on collected information from clients.

We investigate recently proposed federated learning algorithms as shown in Table 1. Most of them improve effectiveness or efficiency by changing only one or several workflow

Method	Modification of Steps				Platform
	Step 1	Step 2	Step 3	Step 4	
FedProx (Li et al., 2020a)		✓		✓	TensorFlow
q-FFL (Li et al., 2020b)		✓		✓	TensorFlow
IFCA (Ghosh et al., 2020)	✓		✓	✓	PyTorch
PowerOfChoice (Cho et al., 2020)	✓				NA
FedDyn (Acar et al., 2021)		✓		✓	PyTorch
Ditto (Li et al., 2021b)	✓	✓	✓	✓	TensorFlow

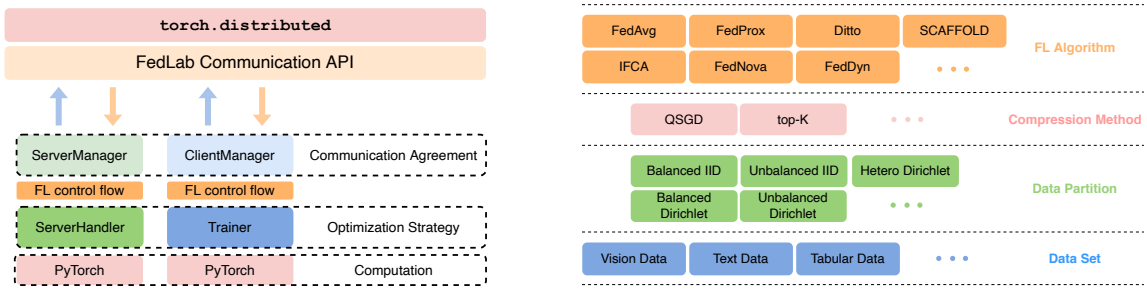
Table 1: The modification of steps for recently published FL algorithms.

steps based on different motivations. This indicates that the implementations of many FL algorithms only require modification on several components of the common workflow, without the necessity of repetitive implementation on basic FL workflows.

To relieve the burden of implementing FL algorithms and to free researchers from the burden of the repetitive implementation of basic FL settings, we develop a highly customizable framework `FedLab` in this work. `FedLab` provides the necessary modules for FL simulation, including communication, model optimization, data partition, and other functional supports. Users can build an FL simulation environment with customized modules using `FedLab` in the way of playing with LEGO bricks.

2. Framework Overview

In this section, we introduce the overview of the proposed library, including the major features, the code usage pipelines, and a discussion of `FedLab`'s impacts on the FL community. For a better understanding, the architecture of `FedLab` is shown in Figure 1(a), and the functional supports are shown in Figure 1(b).



(a) Framework architecture

(b) Functional supports

Figure 1: Overview of the `FedLab` framework

2.1 Features

Communication. For best compatibility with interfaces of PyTorch (Paszke et al., 2019), communication APIs of `FedLab` are built for tensor communication. We provide reliable `send`/`recv` functions for point-to-point communication, while all details of message packaging

and transmission are transparent to users. Users can use communication APIs provided by `FedLab` without knowing the information processing and transmission mechanism.

Communication Compression. Based on the point-to-point communication module, the communication compression algorithms could be easily implemented. A sender could conduct tensor compression before calling the `send` function. Then, the receiver could conduct the corresponding decompression procedure. Additionally, we implement the baseline quantification compressor (Alistarh et al., 2017) and sparsification (Shi et al., 2019) compressor in FL, as shown in *Compression Method* of Figure 1(b).

Communication Agreement. The flexibility of communication management in `FedLab` is given by `NetworkManager` module, which offers users customizable interfaces of communication agreement. Besides, we provide basic *communication pattern*s, namely synchronous and asynchronous FL. Meanwhile, `FedLab` standardizes the implementation of FL communication agreements, and it is open to users' customization needs. Users can define the communication flow between the server and clients for advanced algorithm development.

Federated Optimization. `Trainer`/`ServerHandler` in `FedLab` takes charge of FL optimization procedure on client/server respectively. `Trainer` manages local datasets and performs PyTorch training process; `ServerHandler` works as a computation backend of the server, taking charge of the model aggregation. To demonstrate the usage of `Trainer`, `FedLab` provides `SGDClientTrainer` as a standard implementation of `Trainer` for users, and it can be used as the default `Trainer` in many tasks. Additionally, `FedLab` provides reproduction of different algorithms for uses, as shown in *the FL Algorithm* part in Figure 1(b).

Data Partition & Datasets. The non-IID of data is a key challenge of FL, while realistic Non-IID datasets are not always accessible to researchers. Researchers tend to manually create Non-IID data partitions in the experiment environment (Caldas et al., 2018). For users' convenience, `FedLab` offers `DataPartitioner` for FL data partition. A series of data partition schemes for both IID and Non-IID from different data distribution settings (Yurochkin et al., 2019; Acar et al., 2021; Caldas et al., 2018; Li et al., 2021a) are already provided, including more than 12 data partition schemes for 15 datasets, as shown in *Data Partition* and *Dataset* of Figure 1(b).

Scalability. To meet the requirements of simulation for different scales and simulation acceleration, we design three simulation schemes: *Standalone*, *Cross-process*, and *Hierarchical*. *Standalone* uses `SerialClientTrainer`, allowing one single process to simulate multiple clients sequentially with limited computing resources. *Cross-process* allows larger-scale FL simulation with multiple processes parallelly across computers in the same LAN. Each computer could simulate an arbitrary number of clients' calculation tasks depending on hardware resources. *Hierarchical* provides a communication module `Scheduler`, which enables processes in different Local Area Networks (LANs) to communicate with the global server in the Wide Area Network (WAN).

2.2 Pipelines

Building an FL system with `FedLab` includes two parts. The first part is the definition of a communication agreement. The prototypes of synchronous and asynchronous communication patterns have been implemented for users already. With effortless modification on `NetworkManager` of client/server, users can fulfill the requirements of specific communication

agreements. The second part is `ServerHandler`/`Trainer`, which represents the FL optimization procedure on the server/client. The flexibility of `FedLab` is embodied in practical usages that the user can customize only one or two target components in FL workflow while relying on defaults for the remaining parts. Code examples are shown in Listing 1.

```
# ==== Server Example
smodel = AlexNet()
shandler = ServerHandler(smodel) # Optimization part
snetwork = DistNetwork((server_ip, server_port), world_size, server_rank) # Configuration
smanager = ServerManager(handler, network) # Communication part
smanager.run()

# ==== Client Example
cmodel = AlexNet()
ctrainer = Trainer(cmodel, train_loader, optimizer, criterion) # Optimization part
cnetwork = DistNetwork((server_ip, server_port), world_size, client_rank) # Configuration
cmanager = ClientManager(trainer, network)   # Communication part
cmanager.run()
```

Listing 1: Code examples for server and client

2.3 Impacts

The impacts of `FedLab` on the FL community can be broadly summarized as follows:

Flexible FL Framework. Via the flexible APIs and highly customized modules in `FedLab`, researchers can easily verify their research ideas with simulation acceleration and implementation flexibility.

Comprehensive Learning Materials. Through the provided comprehensive demos and tutorials for FL learners, `FedLab` could support the development of the FL community by helping both beginners and experts.

Integrated Baseline Algorithms and Benchmarks. Researchers can easily conduct intensive comparison experiments based on benchmark datasets and baseline algorithms integrated in `FedLab`.

For the most benefits of the machine learning community, `FedLab` is built on the most popular ML framework PyTorch (according to a study on HuggingFace and PageswithCode [1]), instead of other existing parameter-server frameworks (Chen et al., 2015; Abadi et al., 2016; Jiang et al., 2018). Moreover, a number of federated learning models (Zhang et al., 2022b,a; Sultana, 2022) have been built upon `FedLab`.

3. Conclusion and Future Work

In this paper, we introduce a flexible and lightweight FL framework `FedLab`. `FedLab` provides necessary FL modules, reproduction of FL methods for use, as well as detailed documentation with tutorials online. Thus, `FedLab` can make it painless for researchers to verify their ideas in any stage of FL. Future work directions include providing more FL algorithms and utility experiment toolkits in `FedLab`.

1. `https://www.assemblyai.com/blog/pytorch-vs-tensorflow-in-2022/`

Acknowledgments

This work was partially supported by the National Key Research and Development Program of China (No. 2018AAA0100204), a key program of fundamental research from Shenzhen Science and Technology Innovation Commission (No. JCYJ20200109113403826), the Major Key Project of PCL (No. PCL2021A06), an Open Research Project of Zhejiang Lab (NO.2022RC0AB04), and Guangdong Provincial Key Laboratory of Novel Security Intelligence Technologies (No. 2022B1212010005).

References

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, pages 265–283, 2016.

Durmus Alp Emre Acar, Yue Zhao, Ramon Matas Navarro, Matthew Mattina, Paul N. Whatmough, and Venkatesh Saligrama. Federated learning based on dynamic regularization. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

Dan Alistarh, Demjan Grubic, Jerry Li, Ryota Tomioka, and Milan Vojnovic. Qsgd: Communication-efficient sgd via gradient quantization and encoding. *Advances in neural information processing systems*, 30, 2017.

Theodora S. Brisimi, Ruidi Chen, Theofanie Mela, Alex Olshevsky, Ioannis Ch. Paschalidis, and Wei Shi. Federated learning of predictive models from federated electronic health records. *Int. J. Medical Informatics*, 112:59–67, 2018.

David Byrd and Antigoni Polychroniadou. Differentially private secure multi-party computation for federated learning in financial applications. *CoRR*, abs/2010.05867, 2020.

Sebastian Caldas, Peter Wu, Tian Li, Jakub Konečný, H. Brendan McMahan, Virginia Smith, and Ameet Talwalkar. LEAF: A benchmark for federated settings. *CoRR*, abs/1812.01097, 2018.

Tianqi Chen, Mu Li, Yutian Li, Min Lin, Naiyan Wang, Minjie Wang, Tianjun Xiao, Bing Xu, Chiyuan Zhang, and Zheng Zhang. Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems. *arXiv preprint arXiv:1512.01274*, 2015.

Yae Jee Cho, Jianyu Wang, and Gauri Joshi. Client selection in federated learning: Convergence analysis and power-of-choice selection strategies. *CoRR*, abs/2010.01243, 2020.

Avishek Ghosh, Jichan Chung, Dong Yin, and Kannan Ramchandran. An efficient framework for clustered federated learning. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 19586–19597. Curran Associates, Inc., 2020.

Jie Jiang, Lele Yu, Jiawei Jiang, Yuhong Liu, and Bin Cui. Angel: a new large-scale machine learning system. *National Science Review*, 5(2):216–236, 2018.

Yuang Jiang, Shiqiang Wang, Bong-Jun Ko, Wei-Han Lee, and Leandros Tassiulas. Model pruning enables efficient federated learning on edge devices. *CoRR*, abs/1909.12326, 2019.

Qinbin Li, Yiqun Diao, Quan Chen, and Bingsheng He. Federated learning on non-iid data silos: An experimental study. *CoRR*, abs/2102.02079, 2021a.

Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *Proceedings of Machine Learning and Systems*, 2:429–450, 2020a.

Tian Li, Maziar Sanjabi, Ahmad Beirami, and Virginia Smith. Fair resource allocation in federated learning. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020b.

Tian Li, Shengyuan Hu, Ahmad Beirami, and Virginia Smith. Ditto: Fair and robust federated learning through personalization. In *International Conference on Machine Learning*, pages 6357–6368. PMLR, 2021b.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *AISTATS*, volume 54 of *Proceedings of Machine Learning Research*, pages 1273–1282. PMLR, 2017.

Chuizheng Meng, Sirisha Rambhatla, and Yan Liu. Cross-node federated graph neural network for spatio-temporal data modeling. *CoRR*, abs/2106.05223, 2021.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32:8026–8037, 2019.

Shaohuai Shi, Xiaowen Chu, Ka Chun Cheung, and Simon See. Understanding top-k sparsification in distributed deep learning. *arXiv preprint arXiv:1911.08772*, 2019.

Khadija Sultana. *Elastic Optimization for Stragglers in Edge Federated Learning*. PhD thesis, Victoria University, 2022.

Jie Xu, Benjamin S. Glicksberg, Chang Su, Peter B. Walker, Jiang Bian, and Fei Wang. Federated learning for healthcare informatics. *J. Heal. Informatics Res.*, 5(1):1–19, 2021.

Mikhail Yurochkin, Mayank Agarwal, Soumya Ghosh, Kristjan H. Greenewald, Trong Nghia Hoang, and Yasaman Khazaeni. Bayesian nonparametric federated learning of neural networks. In *ICML*, volume 97 of *Proceedings of Machine Learning Research*, pages 7252–7261. PMLR, 2019.

Tao Zhang, Shaojing Yang, Anxiao Song, Guangxia Li, and Xuewen Dong. Dual adversarial federated learning on non-iid data. In *Knowledge Science, Engineering and Management: 15th International Conference, KSEM 2022, Singapore, August 6–8, 2022, Proceedings, Part III*, pages 233–246. Springer, 2022a.

Zhuo Zhang, Xiangjing Hu, Lizhen Qu, Qifan Wang, and Zenglin Xu. Federated model decomposition with private vocabulary for text classification. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 6413–6425, 2022b.