Poster: Asynchronous Federated Learning Library and Benchmark with AFL-Lib

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

Shuyuan Li*
City University of Hong Kong
shuyuan.li@cityu.edu.hk

Zimu Zhou
City University of Hong Kong
zimuzhou@cityu.edu.hk

Shuo Kang
SKLCCSE, Beihang University
kangshuo@buaa.edu.cn

Yiming Ma
SKLCCSE, Beihang University
ma.yiming@buaa.edu.cn

Yongxin Tong
SKLCCSE, Beihang University
yxtong@buaa.edu.cn

Abstract

Asynchronous Federated Learning (AFL) emerges as a practical paradigm for collaborative model training across IoT devices with heterogeneous compute capabilities, network bandwidth, and online availability. Yet AFL research still lacks a unified, easy-to-use platform for reproducible experimentation under such system configurations. We present AFL-Lib, the first open-source library and benchmark designed for AFL. It allows researchers to flexibly configure device types, network conditions, and availability patterns to emulate the system heterogeneity that gives rise to model staleness, a key factor affecting AFL algorithm design. Beyond system-level settings, AFL-Lib supports plug-in modules for personalized and multi-modal federated training, enabling exploration of the interplay between system and data heterogeneity. AFL-Lib implements 10 state-of-the-art AFL algorithms and 4 synchronous baselines, and integrates 12 datasets spanning image, text, and sensor. We will continue to expand AFL-Lib with new algorithms, datasets, and features to support ongoing AFL research. All code and data are publicly available at https://github.com/boyi-liu/AFL-Lib.

CCS Concepts

• **Computing methodologies** → **Learning paradigms**.

Keywords

Asynchronous Federated Learning; System Heterogeneity; Data Heterogeneity

ACM Reference Format:
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 *The 31st Annual International Conference on Mobile Computing and Networking (ACM MOBICOM '25), November 4–8, 2025, Hong Kong, China.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3680207.3765660

1 Introduction

Asynchronous Federated Learning (AFL) [6] is an emerging paradigm for privacy-preserving and communication-efficient model training across IoT devices. Unlike synchronous FL [2], AFL allows devices to send updates independently, avoiding delays caused by slow or unavailable clients. This makes AFL well-suited for IoT environments, where system heterogeneity is prevalent due to variations in compute power, network bandwidth, and device availability [4].

While AFL addresses inefficiencies in synchronous FL, existing research lacks a unified benchmark that combines algorithmic design with *system configurations*. Prior AFL studies and general-purposed FL frameworks abstract asynchrony as *arbitrary staleness patterns*, without linking them to the underlying system-level causes. This limits the end-to-end evaluation and practical development of AFL algorithms for real-world IoT deployments.

To bridge this gap, we present AFL-Lib, the first open-source library and benchmark tailored for AFL. AFL-Lib enables fine-grained configuration of system heterogeneity, allowing researchers to simulate diverse compute capabilities (*e.g.*, Jetson TX2, Jetson Nano, Raspberry Pi), communication speeds (*e.g.*, Wi-Fi, 4G, 5G), and device availability (via customizable dropout schedules). These configurations can be specified with just a few lines of code.

*Equal contribution.

ACM MOBICOM '25, November 4–8, 2025, Hong Kong, China

ACM ISBN 979-8-4007-1129-9/2025/11
https://doi.org/10.1145/3680207.3765660

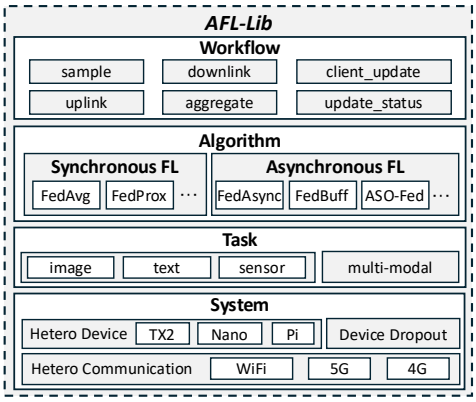

Figure 1: AFL-Lib overview.

Beyond system-level realism, AFL-Lib includes support for *personalization* [1] and *multi-modality* [3], two underexplored but critical aspects of AFL for IoT applications. We implement AFL-Lib with a sequential device simulation pipeline that runs efficiently on a single GPU, lowering the barrier for reproducible experimentation.

The main contributions of AFL-Lib are:

- Flexible configuration of device, communication, and availability heterogeneity to simulate realistic system-induced staleness in AFL.
- Support for personalization and multi-modality to study the interplay between system and data heterogeneity.
- Integration of 10 leading AFL algorithms, 4 synchronous baselines, and 12 datasets across image, text, sensor, and multi-modal domains.

2 AFL-Lib

2.1 Overview

AFL-Lib simulates asynchronous federated learning (AFL) in the classic client-server architecture. Whenever a client completes local training, it immediately uploads its update. The server aggregates the update into the global model and returns the refreshed model to the client. No round-level client synchronization is required.

AFL-Lib decomposes the AFL workflow into six modular stages (see Fig. 1). (1) `sample`: Select a client and mark it as sampled. (2) `downlink`: Transmit the current global model to the selected client. (3) `client_update`: Perform local training with the specified compute capabilities. (4) `uplink`: Upload the model update to the server with the given communication budget. (5) `aggregate`: Integrate the received update into the global model using user-defined model aggregation strategies. (6) `update_status`: update the client's state (*e.g.*, clients' staleness) before the next iteration. Each stage is modular and extensible via Python interfaces, allowing researchers to easily modify one component without rewriting the entire pipeline. AFL-Lib adopts a sequential simulation strategy that runs clients one at a time on a single GPU, while preserving the logical timing and asynchrony of real deployments. This design ensures reproducibility and lowers the barrier to experimentation on common GPU platforms.

Beyond the modular architecture, AFL-Lib offers the following features.

- *System heterogeneity configuration.* Researchers can assign per-client compute profiles, network conditions, and availability to model system-induced staleness.
- *Advanced support for data heterogeneity*. AFL-Lib provides built-in modules for personalized and multi-modal model training, facilitating research on the interaction between system- and data-level heterogeneity.

2.2 Key Features

System Heterogeneity Configuration. Instead of *assigning* staleness values at random, AFL-Lib *derives* staleness from its three system-level causes: compute latency, network latency, and device availability.

- *Device type*: Profiles for three representative IoT devices (Jetson TX2, Jetson Nano, and Raspberry Pi) that capture per-epoch training time based on real-world benchmarking data [5].
- *Communication speed*: Define uplink/downlink bandwidths based on Wi-Fi, 4G, and 5G networks, which translate into transmission delays during the `downlink` and `uplink` stages.
- *Device availability*: Support probabilistic dropout where clients may temporarily go offline, mimicking power-off or intermittent connectivity.

These factors can be configured by editing a YAML file below.

```yaml
dev: # device type config
↪dev_prop: 1 1 1 # Jetson TX2, Jetson Nano, Raspberry Pi

comm: # communication speed config
↪comm_prop: 1 1 1 # WiFi, 4G, 5G

dropout: # device availability config
↪drop_prob: 0.1
```

Interplay with Data Heterogeneity. While AFL is primarily designed to address staleness caused by *system heterogeneity*, practical IoT deployments must also tackle *data heterogeneity*, where clients hold non-IID and multi-modal data. To support research at the intersection of system and data challenges, AFL-Lib includes native support for two key data-centric features: *personalization* and *multi-modality*.

- *Personalization.* AFL-Lib implements four approaches, architecture-, regularization-, mask-, and clustering-based personalization. Each can be activated via minimal code changes. For example, architecture-based personalization requires a single variable (`p_keys`) to define which model components remain client-specific.
- *Multi-modality.* AFL-Lib provides built-in support for multi-modal datasets spanning vision, text, and sensor modalities. It includes flexible fusion strategies such as intermediate and late fusion, allowing users to specify how modalities interact during training.

These features enable researchers to study how personalization and multi-modal learning behave under asynchronous conditions, an area largely unexplored in prior AFL work.

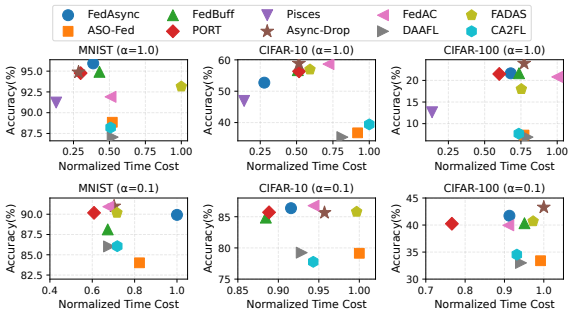

Figure 2: Accuracy v.s. normalized convergence time of 10 AFL algorithms. Upper: w/o personalization; Lower: with personalization.

3 Benchmarks

3.1 Algorithm and Dataset Integration

AFL-Lib integrates a broad and diverse set of algorithms.

- *Synchronous FL:* FedAvg, FedProx, SCAFFOLD, MOON.
- *Asynchronous FL:* FedAsync, FedBuff, ASO-Fed, PORT, Pisces, AsyncDrop, FedAC, DAAFL, FADAS, CA2FL.

To our knowledge, AFL-Lib is the first platform to support 10 AFL methods under a common interface, enabling reproducible comparisons across heterogeneous settings.

In addition, AFL-Lib integrates 10 datasets spanning image (MNIST, EMNIST, FashionMNIST, CIFAR-10, CIFAR-100, SVHN, TinyImageNet), text (AGNews, ShakeSpeare), sensor (IMU, HARBox), and multi-modal domains (CREMAD), providing representative tasks for IoT-centric evaluation.

3.2 Micro-Benchmark Study

We benchmark all 10 AFL algorithms in AFL-Lib on MNIST, CIFAR-10, and CIFAR-100 under Dirichlet non-IID ($\alpha = 1.0$), tracking test accuracy and normalized convergence time. Personalization is tested with client-specific heads and $\alpha = 0.1$. An MLP is applied to MNIST, and a CNN is applied to CIFAR-10 and CIFAR-100. We model device heterogeneity using a 1:1:1 ratio of Jetson TX2, Jetson Nano and Raspberry Pi profiles, and network heterogeneity using Wi-Fi, 4G and 5G configurations in the same proportion. A summary is reported in Fig. 2.

We observe that buffering-based methods (*e.g.*, FedBuff, PORT, Pisces) achieve faster convergence under non-IID conditions, while dropout-based algorithms (*e.g.*, AsyncDrop) are more robust to high staleness and personalization.

4 Conclusion

We present AFL-Lib, the first open-source library and benchmark dedicated to asynchronous federated learning. By modeling system heterogeneity through configurable compute, communication, and availability profiles, and supporting data heterogeneity via personalization and multi-modality, AFL-Lib enables end-to-end, reproducible AFL research. With 10 AFL algorithms, 12 datasets, and modular simulation support, AFL-Lib provides a practical and extensible platform for evaluating and advancing AFL methods. We will continue to expand the library with new algorithms, datasets, and features to support the growing needs of the community.

Acknowledgments

This work was partially supported by National Science Foundation of China (NSFC) (Grant Nos. 62425202, U21A20516, 62336003), the Beijing Natural Science Foundation (Z230001), the Fundamental Research Funds for the Central Universities No. JK2024-03, the Didi Collaborative Research Program and the State Key Laboratory of Complex & Critical Software Environment (SKLCCSE). Zimu Zhou's research is supported by CityU APRC grant (No. 9610633). Zimu Zhou and Yongxin Tong are the corresponding authors.

References

[1] Boyi Liu, Yiming Ma, Zimu Zhou, Yexuan Shi, Shuyuan Li, and Yongxin Tong. 2024. CASA: Clustered Federated Learning with Asynchronous Clients. In *KDD*. 1851–1862.

[2] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *AISTATS*. 1273–1282.

[3] Xiaomin Ouyang, Zhiyuan Xie, Heming Fu, Sitong Cheng, Li Pan, Neiwen Ling, Guoliang Xing, Jiayu Zhou, and Jianwei Huang. 2023. Harmony: Heterogeneous multi-modal federated learning through disentangled model training. In *MobiSys*. 530–543.

[4] Kilian Pfeiffer, Martin Rapp, Ramin Khalili, and Jörg Henkel. 2023. Federated learning for computationally constrained heterogeneous devices: A survey. *Comput. Surveys* 55, 14s (2023), 1–27.

[5] Ahmet Ali Süzen, Burhan Duman, and Betül Şen. 2020. Benchmark analysis of jetson tx2, jetson nano and raspberry pi using deep-cnn. In *HORA*. 1–5.

[6] Cong Xie, Sanmi Koyejo, and Indranil Gupta. 2019. Asynchronous federated optimization. *arXiv preprint arXiv:1903.03934* (2019).