FS-Real: A Real-World Cross-Device Federated Learning Platform

Dawei Gao*, Daoyuan Chen*, Zitao Li, Yuexiang Xie, Xuchen Pan,
Yaliang Li, Bolin Ding, Jingren Zhou
Alibaba Group

ABSTRACT

Federated learning (FL) is a general distributed machine learning paradigm that provides solutions for tasks where data cannot be shared directly. Due to the difficulties in communication management and heterogeneity of distributed data and devices, initiating and using an FL algorithm for real-world cross-device scenarios requires significant repetitive effort but may not be transferable to similar projects. To reduce the effort required for developing and deploying FL algorithms, we present FS-Real, an open-source FL platform designed to address the need of a general and efficient infrastructure for real-world cross-device FL. In this paper, we introduce the key components of FS-Real and demonstrate that FS-Real has the following capabilities: 1) reducing the programming burden of FL algorithm development with plug-and-play and adaptable runtimes on Android and other Internet of Things (IoT) devices; 2) handling a large number of heterogeneous devices efficiently and robustly with our communication management components; 3) supporting a wide range of advanced FL algorithms with flexible configuration and extension; 4) alleviating the costs and efforts for deployment, evaluation, simulation, and performance optimization of FL algorithms with automatized tool kits.

PVLDB Reference Format:
Dawei Gao, Daoyuan Chen, Zitao Li, Yuexiang Xie, Xuchen Pan, Yaliang Li, Bolin Ding, and Jingren Zhou. FS-Real: A Real-World Cross-Device Federated Learning Platform. PVLDB, xx(x): XXX-XXX, 2023.
doi:XX.XX/XXX.XX

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/alibaba/FederatedScope/tree/FSreal.

1 INTRODUCTION

Federated Learning (FL) is widely recognized as a promising learning paradigm for preserving data privacy and sovereignty by distributing the machine learning training process across multiple devices, and requiring the devices to only share model updates instead of raw local data [7]. This design can also achieve high communication efficiency by enabling devices to share model updates *only after every few rounds of local training*. Further, by incorporating diverse data sources, FL has great potential to improve the performance of machine learning models, in terms of better adaptation to a wide range of scenarios and more accurate and personalized results [4].

Nonetheless, the distributed nature of FL, coupled with data heterogeneity (data following different distributions) and device heterogeneity (learning is conducted on various software and hardware environments), results in greater complexity compared to centralized learning, and presents considerable challenges in the development, evaluation, and deployment of FL algorithms. In a cross-device context, there are typically thousands to millions of devices that have limited hardware computation capacities and unstable network connections. Thus, implementing cross-device FL algorithms demands substantial effort, including the development of local training modules for diverse edge devices, aggregation of updates from numerous concurrent communications, management of time-delayed and drop-out clients, and evaluation of model and system performance [3]. Certain components of existing FL implementations, such as inter-party communication, often serve similar purposes but are not reusable across different projects or systems involving diverse device types and device scales. Moreover, due to the absence of dedicated tools for managing or simulating a multitude of heterogeneous devices, the evaluation of existing FL algorithms (especially in terms of system and model performance) may lead to discrepancies between reported research findings and real-world outcomes. Overcoming these challenges is crucial for the widespread adoption and successful implementation of cross-device FL in real-world applications.

To address these challenges, we propose a comprehensive, efficient, and flexible platform for the development, simulation, and deployment of FL algorithms, named FS-Real. The proposed platform, derived from an event-driven FL framework named FederatedScope [10], achieves significant progress in tackling the aforementioned challenges in cross-device FL. Specifically, FS-Real offers encapsulated interfaces that conceal low-level details from FL algorithm designers and users on both the server and client sides, and supports adaptation across various types of devices. Moreover, FS-Real is equipped to handle the complexities and large communication volumes associated with cross-device FL settings, making that FS-Real can process messages up to 3.9x more efficiently than FS and other competitive FL frameworks designed toward scaling [5], and can support 100,000 scale devices as shown in [2]. Additionally, the proposed FS-Real can be flexibly integrated with various advanced FL techniques such as personalization, compression and asynchronous aggregation to satisfy real-world requirements, and has been plugged with an easy-to-use GUI and a monitoring module for helping users to configure, track and analyze the FL process. In summary, the proposed FS-Real provides a comprehensive solution that streamlines the FL research and development processes by simplifying implementations and providing a standardized platform for comparing different cases across various FL scenarios.

In this demo, we first introduce the core components in FS-Real, including: (1) Client runtimes, built on various devices operating systems (Android, AliOS, and Linux on embedded devices) and

*Co-first authors.

Proceedings of the VLDB Endowment, Vol. xx, No. x ISSN 2150-8097.
doi:XX.XX/XXX.XX

MNN [6] APIs, provide customizable data preprocessing and local training functions. This frees users from handling underlying programming interfaces. (2) An exceptionally efficient message handling and model aggregation module, enables the server to manage cross-device FL tasks involving millions of clients. (3) A collection of flexible programming interfaces that allow users to easily customize the key FL processes, such as training, communication, and aggregation. With these interfaces, a wide range of advanced FL algorithms can be supported such as asynchronous training, model compression, and personalized FL [1]. (4) A user-friendly simulation platform that can be initiated with a single command, and can faithfully emulate the complete FL training process involving a large number of clients with configurable heterogeneous devices.

After walking through these components, we demonstrate how to use FS-Real with three distinct scenarios. • In the first scenario, we illustrate how users can conduct FL processes using our Android runtime and server-side support on a vision dataset. After completing a few configuration steps on mobile devices, users can seamlessly participate in the FL process. The FL coordinator can also monitor various aspects of the process, including device hardware distribution, system performance, and model performance. • In the second scenario, we demonstrate how to initiate our simulation platform and configure it easily to incorporate advanced FL techniques on a textual dataset, achieved by simply modifying a few values in configuration files. • In the third scenario, we showcase a real-world FL application running on cars with speech data, backed by the FS-Real platform.

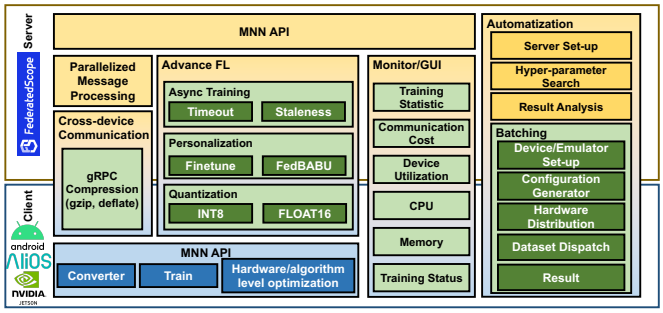


Figure 1: Overall architecture of FS-Real. Gold blocks are on the server side; blue blocks are on the client side; green blocks involve functions on both server and client side.

2 FS-REAL: ARCHITECTURE IMPROVEMENT

In this section, we summarize the major components of FS-Real as illustrated in Figure 1. We refer readers to [2] for more details about the system design and implementations.

Specifically, the server side of FS-Real is adapted from an event-driven FL system, FederatedScope (FS) [10]. Although FS has demonstrated its potential to support a wide range of FL tasks, it falls short in addressing real-world cross-device FL challenges stated below: (i) Being primarily implemented in Python, FS does not inherently account for the computational limitations and implementation heterogeneity of edge devices with their specific hardware and operating systems. (ii) FS struggles when the number of clients is more than 10,000 in distributed mode, a minimum requirement for cross-device FL. (iii) FS's server side does not sufficiently consider robustness and efficiency in dealing with resource-limited and connection-spotty clients. To address these shortcomings, we refactor and enhance several key components of FS and build matched runtimes for Android and IoT devices, resulting in FS-Real as a next-generation FL system tailored for real-world cross-device FL.

FS-Real runtime: Towards easing programming local training and optimizing system performance on heterogeneous devices. We provide plug-and-play runtimes that can be executed on various types of devices, including Android, NVIDIA Jetson and AliOS for intelligent cars. Users can effortlessly customize the runtime to manage their own FL tasks by defining the data paths and implementing the optional data preprocessing functions, without having to start from scratch with the underlying operating system APIs. The training and inference components of the runtimes are developed in C++ and utilize MNN [6], which offers extensible interfaces for common yet fundamental learning behaviors to support advanced FL algorithms such as personalization [1]. The runtimes also provide hardware-level optimization on different devices' hardware structures to ensure efficient memory and time usage. In response to this change, the server-side implementation has also been modified and optimized to support MNN accordingly, where the implementations in Python are well-supported with the help of conversion among MNN, ONNX[1], and PyTorch models. Communication between the participating devices and the FL server is built on gRPC, which provides efficient cross-device message transfer, flexible message sharing, and lossless compression tools to reduce network traffic.

Parallelized message processing: Towards providing more efficient and flexible FL communication. Unlike cross-silo FL with only a few participated clients, the server in cross-device FL must manage thousands or even millions of heterogeneous devices during the training process. To accommodate this challenging demand, our system is equipped with a multi-process parallel message sending, receiving and processing mechanism on the server side, ensuring that the server can handle numerous model updates concurrently. Moreover, this new message-handling mechanism can be configured to manage late or asynchronous messages effectively, which helps to deal with slow and network-spotty devices. Some lossless compression methods are also embedded in this message-processing component, including Gzip and Deflate.

Client selection: Towards ensuring robustness and efficiency at the client level. In cross-device FL, clients may have varying computational power and network accessibility during the training process, making drop-outs and delays quite common. However, only a subset of clients is required to participate in one training round. To address this, we provide a component that enables the server to not only track the status of clients in each iteration but also actively select the subset of clients to be involved in the next iteration based on their performance records. The system can be configured to select a slightly larger portion of users than the preferred aggregation number (over-selection) to ensure there are sufficient in-time updates even if some clients drop out. Our system can automatically

[1] https://onnx.ai/

adjust the timeout duration based on the responsiveness of clients, making the system adaptable under various network conditions. If an FL application demands high efficiency and has low delay tolerance, the server can also be configured to favor reliable clients, trading potential bias for improved training efficiency.

High-fidelity simulation platform: Towards making the first step of research and development easier. Prototyping FL solutions that involve recruiting or collecting a large number of heterogeneous devices can be expensive and resource-intensive. To address this challenge, our system also offers a simulation platform that can support high-fidelity and efficient simulation for the FL process of thousands upon thousands heterogeneous devices using only a few servers. The resources of participating client devices (including computational power and network accessibility), hyperparameters on each device and data distribution are all configurable. Users can define a resource distribution from which resource configurations are sampled, creating an application context that closely resembles real-world scenarios. In addition, the simulation platform can be started with a single command and clear step-back-step feedback, making the user's learning curve shallow.

3 DEMONSTRATION

In this section, we showcase how the proposed system serves as a robust infrastructure in three scenarios, ranging from the most basic use case (running FL with our vanilla Android runtime) to more advanced techniques (utilizing our simulation platform with various FL performance enhancement methods) and practical applications (supporting a car-related app).

3.1 Running with the default Android runtime

Figure 2a demonstrates the basic steps and expected outputs of running our vanilla Android runtime.

Before FL training. There are default data preprocessing and training functions (based on FedAvg [7]) implemented already in the vanilla Android runtime. Thus, after compiling and installing the runtime, the only thing users need to do before initiating the FL training is to set the training-related parameters (⓪ in Figure 2a). The users can directly adjust those parameters via the GUI of the runtime, by clicking the setting icon in the upper-right corner. The parameters that need to be specified include the ports and IP addresses of the devices and the FL server, the model type, and the training-related numbers (*e.g.*, learning rate and batch size).

During FL training. There are mainly three types of communication between server and device in FS-Real by default: join-in, training, and evaluation. The client can activate the application and click the "CONNECT TO SERVER" button to join the configured FL task (① in Figure 2a). The underlying function then sends a join-in message to the server. If the server validates the eligibility of the client, it will send a confirmation message with an assigned client ID back. When a certain number of clients have joined for the FL tasks, the server will start the FL training by broadcasting the global model to the devices and request for local training. The device will upload an updated model to the server after running configured local training epochs. During the local training phase, training information will be displayed on the screen, including the FL iteration, local training epoch, training loss, and resource usage (② in Figure 2a). The process will repeat a few rounds as specified in the configuration file. The server may request a local evaluation from local devices occasionally (*e.g.*, every 10 rounds) or at the end of the training by sending local evaluation requests to clients. To answer such requests, the device will use a local validation/test set to evaluate the received model, print the results on screen and also send them back to the server (③ in Figure 2a).

Server side monitor. On the server side, an easy-to-use monitor is available to display various types of FL system or model performance information (④ in Figure 2a). The web GUI of the monitor is implemented using WandB [2]. The monitor can generate charts about device hardware distribution upon receiving device join-in messages, update network traffic, and display CPU or memory usage during the training process. It can also show validation/test loss and accuracy after aggregating local evaluation results from the selected devices, including the globally weighted metrics and several fairness-related metrics. To customize the information displayed on the page, users can easily specify which metrics should be evaluated and logged with simple configuration and WandB.

3.2 Running experiments with advanced FL techniques on FS-Real simulation platform

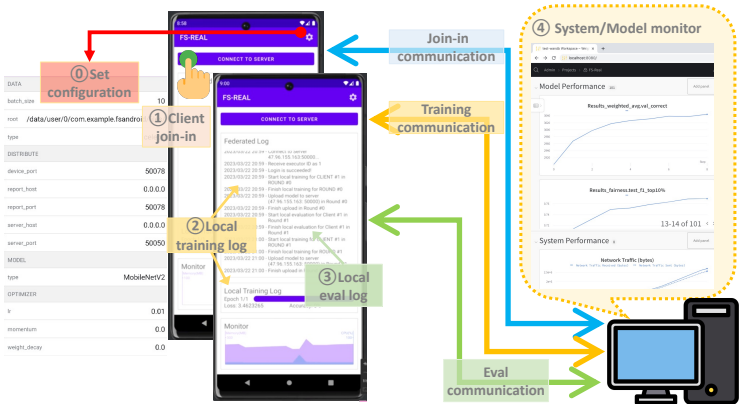
We provide an example of how to utilize our simulation platform for experiments and how to integrate our system with advanced FL features. Figure 2b illustrates the process.

Starting simulation platform. After deploying the system on the server, users can then initiate the simulation with a single command with provided or customized YAML configuration files, with the IP addresses and port number specified as well, as demonstrated in Figure 2b. The information displayed on the terminal indicates the progress of setting up the simulation.

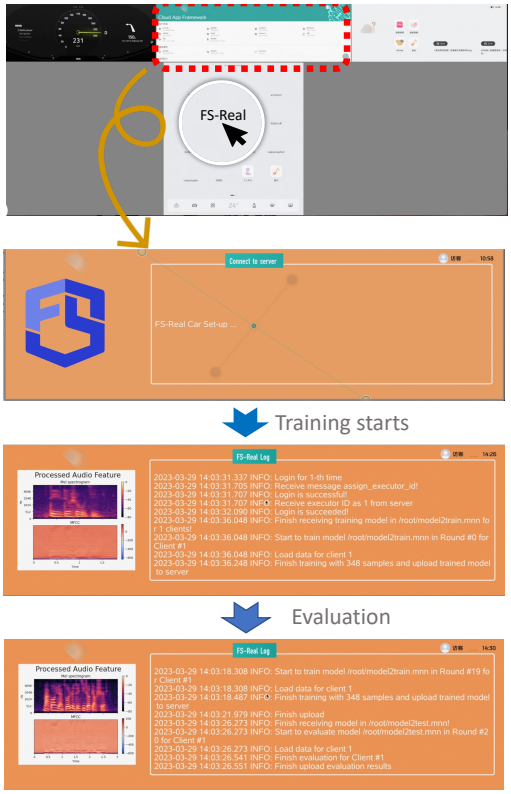
Working with advance FL. As for starting with some advanced FL techniques, FS-Real users can refer to the provided configuration files. The right side of Figure 2b provides some examples of ready-to-use advanced FL algorithms in the FS-Real.

- When the network cost is a main concern, users can turn on the lossless compression function via adding "`grpc_compress`" with "`gzip`" as the method under the "`distribute`" keyword, and/or turn on the lossy compression by further quantizing the model via adding "`mnn_quantization`" with "`int8`" as the target precision.
- If users want to adopt personalization techniques to improve the models on local datasets, FS-Real provides some implementation for reference as well. In the example, we turn on the FedBABU algorithm [9]. Users can specify personalization training hyperparameters, including learning rate, local update step, etc., in the YAML file under the "`personalization`" keyword.
- If users want to improve the system's efficiency by turning on the asynchronous message processing mechanism, they can add the "`asyn`" keyword in the YAML file to use the default FedBuff algorithm [8]. The asynchronous training parameters, such as the "`staleness_toleration`" controlling the tolerable staleness (the

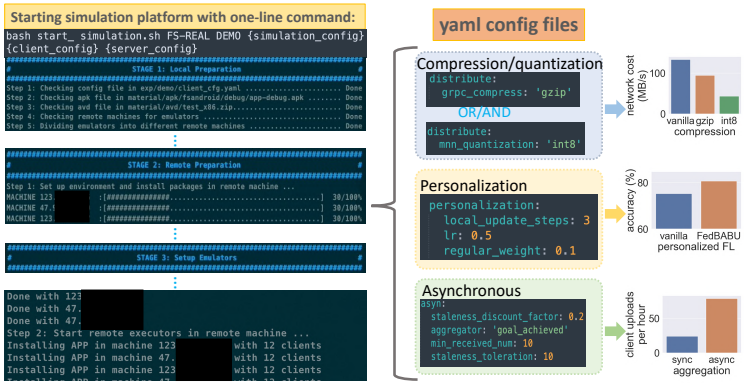
[2] https://wandb.ai/site



(a) Running FS-Real with Android runtime.



(c) A speech recognition application on the car.



(b) Simulation platform with advanced FL techniques.

Figure 2: Three demonstration scenarios of FS-Real.

maximum difference number of FL rounds used in aggregation), can be specified after the "asyn" keyword.

By turning on these advanced FL techniques, the corresponding metrics are expected to be improved. For example, the network cost can be reduced with gzip or quantization; the local testing accuracy can be improved with the personalized models; the client utilization can be improved in the asynchronous mode, as shown in the right part of figure 2b. We also allow users to easily extend their own FL algorithms and introduce new parameters into YAML files.

3.3 Supporting FL with intelligent cars

FS-Real can be easily adapted into other IoT applications running on various OS, including the embedded NVIDIA Jetson system and AliOS system for intelligent cars. Here we show such a successful case of supporting a federated speech recognition task on the AliOS system in Figure 2c. Due to the fact that the FL tasks on IoT devices usually run in the background without a GUI (*e.g.*, as a dynamic library being integrated into third-party applications), here we print the training logs on the car screen to demonstrate that the training follows the same paradigm as running on Android devices and the simulation platform. Thanks to the portability of the MNN and gRPC, FS-Real can greatly save users' efforts on communication management and local training modules when migrating the built-in FL projects onto other types of devices. The compiling, building and packaging are easy-configurable according to the target hardware with automatic optimization on the computation and storage efficiency.

REFERENCES

[1] Daoyuan Chen, Dawei Gao, Weirui Kuang, Yaliang Li, and Bolin Ding. 2022. pFL-Bench: A Comprehensive Benchmark for Personalized Federated Learning. In *NeurIPS'22, Datasets and Benchmarks Track*.

[2] Daoyuan Chen, Dawei Gao, Yuexiang Xie, Xuchen Pan, Zitao Li, Yaliang Li, Bolin Dingand, and Jingren Zhou. 2023. FS-Real: Towards Real-World Cross-Device Federated Learning. *arXiv preprint arXiv:2303.13363* (2023).

[3] Imteaj, Ahmed, et al. 2022. A Survey on Federated Learning for Resource-Constrained IoT Devices. *IEEE Internet of Things Journal* 9, 1 (2022), 1–24.

[4] Kairouz, Peter, et al. 2021. Advances and open problems in federated learning. *Foundations and Trends in Machine Learning* 14, 1–2 (2021), 1–210.

[5] 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 *ICML'22*. 11814–11827.

[6] Lv, Chengfei, et al. 2022. Walle: An End-to-End, General-Purpose, and Large-Scale Production System for Device-Cloud Collaborative Machine Learning. In *OSDI'22*. 249–265.

[7] 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 *AISTATS'17*. 1273–1282.

[8] Nguyen, John, et al. 2022. Federated learning with buffered asynchronous aggregation. In *AISTATS'22*. 3581–3607.

[9] Jaehoon Oh, SangMook Kim, and Se-Young Yun. 2022. FedBABU: Toward Enhanced Representation for Federated Image Classification. In *ICLR'22*.

[10] Yuexiang Xie, Zhen Wang, Dawei Gao, Daoyuan Chen, Liuyi Yao, Weirui Kuang, Yaliang Li, Bolin Ding, and Jingren Zhou. 2023. FederatedScope: A Flexible Federated Learning Platform for Heterogeneity. *PVLDB* 16, 5 (2023), 1059–1072.