FederatedScope: A Comprehensive and Flexible Federated Learning Platform via Message Passing

Yuexiang Xie*, Zhen Wang*, Daoyuan Chen†, Dawei Gao†, Liuyi Yao†, Weirui Kuang‡, Yaliang Li‡†, Bolin Ding‡†, Jingren Zhou
Alibaba Group

ABSTRACT

Although remarkable progress has been made by the existing federated learning (FL) platforms to provide fundamental functionalities for development, these FL platforms cannot well satisfy burgeoning demands from rapidly growing FL tasks in both academia and industry. To fill this gap, in this paper, we propose a novel and comprehensive federated learning platform, named FederatedScope, which is based on a message-oriented framework. Towards more handy and flexible support for various FL tasks, FederatedScope frames an FL course into several rounds of message passing among participants, and allows developers to customize new types of exchanged messages and the corresponding handlers for various FL applications. Compared to the procedural framework, the proposed message-oriented framework is more flexible to express heterogeneous message exchange and the rich behaviors of participants, and provides a unified view for both simulation and deployment. Besides, we also include several functional components in FederatedScope, such as personalization, auto-tuning, and privacy protection, to satisfy the requirements of frontier studies in FL. We conduct a series of experiments on the provided easy-to-use and comprehensive FL benchmarks to validate the correctness and efficiency of FederatedScope. We have released FederatedScope for users on https://github.com/alibaba/FederatedScope to promote research and industrial deployment of federated learning in a variety of real-world applications.

1 INTRODUCTION

As one of the feasible solutions to address the privacy leakage issue when utilizing large amount of real-world data for training, federated learning (FL) [26, 38, 57] has been witnessed rapidly growing popularity from both academia and industry in recent years. Such marvellous progress is inextricably tied to the support of FL platforms, such as TFF [5], LEAF [7], PySyft [63] and FATE [57], which provide fundamental functionalities for developers to get started quickly and focus on building new algorithms and applications. Despite remarkable efforts have been made by the existing FL platforms, there still exist several burgeoning demands from FL research and applications that cannot be easily satisfied by the existing ones. Furthermore, the ever-increasing demands also challenge the extensibility and flexibility of FL platforms, since developers might get tired of repeatedly refactoring for Supporting new functional components.

Motivated by the aforementioned issues, in this paper, we propose FederatedScope, a novel and comprehensive platform for federated learning based on a novel message-oriented framework. Specifically, we frame an FL course from the view of message passing among participants, and allow customizing heterogeneous data exchange and rich behaviors of participants to describe various FL applications. Compared with the existing FL platforms, FederatedScope provides more handy and flexible support for different users, which can be summarized as follows.

Firstly, FederatedScope provides convenient usage for getting started. An FL platform is expected to be friendly and convenient for users who are junior for federated learning. To this end, first of all, FederatedScope guides users to get started via preparing detailed tutorials, documents, and executable examples such as federally training CNNs with vanilla FedAvg [38] on widely-adopted benchmark dataset FEMNIST [7, 10]. Moreover, FederatedScope allows users to utilize customized datasets and models via registering and configuring when building up the standard FL course in real-world applications. Since we decouple the federated behaviors (e.g., coordinating the participants) and training behaviors (e.g., local update) in FederatedScope, developers only need to prepare the customized datasets and models without worrying about the implementation details of the FL process. Besides, benefiting from the message-oriented framework, FederatedScope provides a unified view for both standalone simulation and distributed deployment, which bridges the gap between academia and industry, and saves developers’ efforts in transforming simulation to deployment.

Secondly, FederatedScope provides flexible support for various FL applications. Apart from supporting the standard FL course where participants only exchange homogeneous data (e.g., model parameters), an FL platform needs to be flexible and extendable for various FL tasks in different real-world scenarios. For example, the emerging Federated Graph Learning (FGL) [54, 55, 58] requires heterogeneous data exchange (such as model parameters, gradients, node embeddings, etc.), and Vertical Federated Learning [12, 16, 20] needs to share public keys and exchange encrypted intermediate results among participants. Based on the proposed message-oriented framework, an FL course is framed into multiple rounds of message passing among participants, where the exchanged data (e.g., model parameters) are abstracted as messages and the behaviors of servers and clients (e.g., aggregation and local update) are transformed into handling functions accordingly. In this way, to describe a new FL task that contains heterogeneous data exchange and rich behaviors, FederatedScope naturally allows users to customize the type of exchanged data, together with the

*Co-first authors.
†Equal contribution, listed in alphabetical order.
‡Corresponding authors, email addresses: {yaliang.li, bolin.ding}@alibaba-inc.com
Personalization, natural language processing and graph learning, and provide mental results of the provided benchmark in FederatedScope can model architectures and prepare scripts to conduct experiments. Meanwhile, we implement a ModelZoo that contains widely-used and preprocess the datasets in FederatedScope. Developers can build up an easy-to-use and comprehensive FL benchmark, we collo.

In order to build up an easy-to-use and comprehensive FL benchmark, we collect public datasets for various FL applications, including computer vision, natural language processing and graph learning, and provide a unified interface of dataloader that can automatically download and preprocess the datasets in FederatedScope. Developers can either run experiments on the provided datasets via configuring, or register a customized dataset for conducting fair comparisons. Meanwhile, we implement a ModelZoo that contains widely-used model architectures and prepare scripts to conduct experiments on the benchmark datasets effortlessly. The statistics and experimental results of the provided benchmark in FederatedScope can be found in Section 5. Besides, when conducting experiments with customized datasets and models, users can utilize the auto-tuning component in FederatedScope to search the suitable hyperparameters, and monitor various client-wise and global metrics for further tuning the FL course.

The overall architecture of FederatedScope is shown in Figure 1. In the rest of this paper, we first introduce the infrastructure of FederatedScope in Section 2, including different modules and the design of message-oriented framework. Then, in Section 3, several functional components, e.g., personalization, auto-tuning, and privacy protection, are detailed described. We present the comprehensive FL benchmark in Section 4, and show the experimental results in Section 5 to better understand the characteristics of FederatedScope. Finally, we conclude this paper in Section 6.

2 INFRASTRUCTURE

In this section, we introduce the infrastructure of FederatedScope.

2.1 Preliminary

Federated Learning (FL) [26, 38, 57], a learning paradigm for collaboratively training models from dispersed data without directly sharing private information, involves multiple participants who are willing to contribute their local data and/or computation resources. We use servers to denote the participants that are responsible for coordinating and aggregating while calling others as clients. A training round of an FL course indicates the clients update models locally and send feedbacks to the servers for collaborative optimization, which includes data exchange among servers and clients, such as model parameters, public keys, hyperparameters, etc. These exchanged data are abstracted as messages in FederatedScope, and note that different FL tasks might involve different types of messages. Generally, the goal of federated learning is jointly training global models in a privacy-preserving manner, which satisfies the requirements of privacy protection and achieves outperformance compared to that without collaboration.

In FederatedScope, we modularize an FL procedure into Worker module and Communication module, and design a message-oriented framework for flexible programming.

2.2 Worker Module

The Worker module of FederatedScope consists of Server and Client, which describe the behaviors of the participants in an FL course and is designed towards conveniently expressing rich behaviors.

2.2.1 Client. The behaviors of a client in an FL course can be categorized into federated behaviors and local training behaviors. The federated behaviors contain exchanging various types of messages, such as, sending an application to the server for joining the FL course, and broadcasting the communication address to other clients. The local training behaviors denote that the clients train the local models on the private data, which might vary a lot in different FL applications in terms of fetching data, loading models, optimizing, etc. Considering both flexibility and convenience, we define the basic member attributes of a client as follows:
federated behaviors and training behaviors can be orthogonal. For clients are decoupled with each other, thus the modifications of personalized adaptation, on their data and model independently. Secondly, the training behaviors and federated behaviors of the client-specific operations, such as data augmentation and model for the clients are isolated. And the clients can perform press the rich behaviors of the clients in various FL applications.

Firstly, according to the setting of federated learning, the data and press the rich behaviors of the clients in various FL applications.

Based on the above attributes, developers can conveniently express the rich behaviors of the clients in various FL applications. Firstly, according to the setting of federated learning, the data and the model for the clients are isolated. And the clients can perform client-specific operations, such as data augmentation and model personalized adaptation, on their data and model independently. Secondly, the training behaviors and federated behaviors of the clients are decoupled with each other, thus the modifications of federated behaviors and training behaviors can be orthogonal. For example, developers can adapt the communication frequency without caring about the model architecture, or focus on customizing a new training algorithm for an implemented FL course. What’s more, the communicator can free developers from implementing the low-level communication backend, and provides a unified interface for both standalone and deployment.

2.2.2 Server. The server is required to coordinate the clients and perform aggregation for federal training, such as initializing an FL course and admitting clients to join in, broadcasting models and configurations, aggregating the updated models, etc. To full-fit these requirements, the basic member attributes of a server includes ID, model, aggregator and communicator:

- **ID.** The server’s ID is unique in an FL course and often bound with its communication address.
- **Model.** The model kept in the server is shared among participants for collaboratively training in an FL course.
- **Aggregator.** The server holds an aggregator to perform aggregation. The aggregator takes the clients’ feedback as inputs (e.g., updated models and statistics of training data) and outputs the aggregated model. Similar to the trainer, in order to decouple the aggregating behaviors and federated behaviors of the server, the aggregator only exposes high-level interfaces, such as AGGREGATE, to the server and hides the details of the adopted algorithms.
- **Communicator.** The communicator is used to exchange message, which is similar to the communicator of the client and is introduced in Section 2.3.2.

In some cases, the server holds test data for global evaluation [8], and a sampler for strategically choosing active clients at each training round [29, 51]. Similar to the design of clients, we decouple the aggregating behaviors and federated behaviors to make the server more flexible and extendable, and hide the low-level communication backend through the communicator.

Note that the behaviors of servers and clients are encapsulated as handling functions as subroutines to handle the received messages. Then an FL procedure is composited by various types of exchanged messages and the corresponding handling functions. Such a perspective of framing an FL course is named the message-oriented framework in our study, which is introduced in Section 2.4. Before that, we present the Communication module in detail.

![Diagram](image.png)

*Figure 2: The messages are passed from a client to a server through the communicators.*
2.3 Communication Module

In the Communication module of FederatedScope, we introduce the design of message and communicator.

2.3.1 Message. The message is a general designation of the exchanged data among servers and clients in an FL course, which can be used to indicate model parameters, gradients, training configurations, join in applications, evaluation results, and so on. Since heterogeneous messages are exchanged among servers and clients, we use TYPE to distinguish different types of messages. Besides, as shown in Figure 2, the SENDER and RECEIVER are recorded in the message for tracing and verifying, and the PAYLOAD denotes the contained data for exchanging.

The message is one of the key objects to describe and drive an FL course in the proposed message-oriented framework. On the one hand, developers can define various types of messages according to what kinds of data are expected to be exchanged among participants during an FL course. On the other hand, different behaviors of servers and clients are triggered by receiving different types of messages, such as local training and aggregating, which promotes the federal training process among participants.

2.3.2 Communicator. The communicators are hosted by servers and clients for exchanging messages with each other. The communicator serves as a black box from the perspectives of servers and clients, since it only exposes high-level interfaces such as SEND and RECEIVE to servers and clients while hiding the details of the communication backend. An example of the message passing from a client to a server via communicator is illustrated in Figure 2, from which it can be observed that the client prepares the message and sends it to the server via the communicator without perceiving which communication backend is used.

Supported by the communicator, FederatedScope is able to provide a unified interface for running an FL course with standalone mode (via simulated communicator) and distributed mode (via communicator based on gRPC or other protocols). Such design is helpful for developers who want to transform simulation to deployment, and bridges the gap between academia and industry.

2.4 Message-Oriented Framework

In this section, we introduce how we implement an FL procedure based on a message-oriented framework.

2.4.1 Message passing. In FederatedScope, we frame an FL course from the perspective of message passing and describe an FL course in an event-driven manner rather than a sequential one. To be more specific, to program an FL procedure with the message-oriented framework, developers should first abstract the types of messages that are exchanged among participants, and then transform the behaviors of servers and clients into handling functions as subroutines to handle different types of received messages. Finally, developers can instantiate the participants and initialize the FL course.

A standard FL procedure viewed from the perspective of message passing is shown in Figure 3, which illustrates the characteristics of the message-oriented framework in FederatedScope compared to the procedural programming paradigm. To perform the vanilla FedAvg, the server is expected to handle two types of messages, i.e., handle join for admitting a new client to join in the FL course, and handle the updated models to perform aggregation. As for the clients, they should train model on the local data and return the feedback (e.g., the updated model) when receiving model from the server. After defining these message types and handlers, the FL course can be triggered by the clients sending join to the server. In this way, developers can program an FL produce by focusing on how servers and clients handle different types of messages without caring about the federal staff such as coordinating participants.

2.4.2 Procedural v.s. Message-oriented. We compare the proposed message-oriented framework with the procedural programming paradigm by showing two examples, including a standard FL course...
illustrated in Figure 3 and a practical FL application that needs heterogeneous data exchange and rich behaviors of participants, as shown in Figure 4. From these examples, we summarize the advantages of the message-oriented framework as follows.

(i) **Support flexible expression for heterogeneous data exchange and rich behaviors.** Although the standard FL course shown in Figure 3 only exchanges homogeneous data (i.e., the model parameters) at each training round, real-world FL applications need heterogeneous data change. For example, Federated Graph Learning applied in recommendation systems [54] and financial risk control systems [44] needs to exchange heterogeneous data including model parameters, gradients, node embeddings, etc. And public keys and intermediate results are exchanged in Vertical Federated Learning [12, 16, 20]. Meanwhile, to handle various types of exchanged data, it is necessary for developers to flexibly express rich behaviors of servers and clients. The proposed message-oriented framework satisfies the aforementioned requirements, since servers and clients can be programmed separately and coordinated via message passing.

We demonstrate a practical FL application in Figure 4 to show how to describe heterogeneous message exchange and handling, where clients need to exchange intermediate results at each training round. To achieve this, developers who adopt a procedural programming paradigm are required to describe the complete FL course sequentially and insert the new behaviors into the implemented FL course after carefully positioning. When using the proposed message-oriented framework, to express heterogeneous data change and rich behaviors, developers only need to define the new types of messages and the corresponding handling functions of servers and clients, eliminating the efforts for coordinating participants. For example, developers can add a new type *intermediate results* for clients and specific that clients continue training locally when receiving the intermediate results. Different from using the procedural programming paradigm, developers would not be bothered by positioning the new behavior (i.e., exchanging intermediate results) in the FL course.

(ii) **Provide a unified view for both simulation and deployment.** With the procedural programming paradigm that most existing FL platforms adopted, developers need to describe a static computation graph for an FL course, and coordinate the data exchanging process among servers and clients. Although the procedural programming paradigm provides a straightforward manner for standalone simulation since the whole FL procedure is described globally and sequentially, it can be unsuitable and complicated when promoting distributive deployment, especially in cross-silo scenarios. Consider the situation that several companies aim to jointly train a global model via federated learning, and each company would not like to leak the details of their local behaviors besides the necessary ones (such as the formatted of shared data). For example, a client might only focus on the local training process and ignore how other clients train their models or how the server performs aggregation. The procedural programming paradigm would be hard to handle such situations since the participant are separately and each client might only be aware of the FL course from a local perspective. Different from the procedural programming paradigm, the proposed message-oriented framework allows developers to build up an FL course via describing servers and clients separately, which
provides a unified view for both standalone simulation and deployment on distributed systems. Users can change from standalone simulation to distributed deployment effortlessly.

(iii) **Allow different backends and toolkits for implementing message handlers.** Based on the message-oriented framework, the behaviors of servers and clients are transformed into handling functions as subroutines to handle different types of received messages. Note that these subroutines are independent, and therefore they can be implemented with different backends and toolkits. For example, developers are allowed to handle one type of message via Tensorflow (e.g., local training) and another based on PyTorch (e.g., data augmentation). Such compatibility is important and necessary for the FL platforms to integrate the existing ML algorithms implemented by different backends and to satisfy a variety of demands from various FL applications. Besides, benefiting from such a design, the message-oriented framework is suitable for the device-cloud collaboration scenario, where a computation graph might be decomposed into several parts and run with different software/hardware environments.

In a nutshell, based on the proposed message-oriented framework, FederatedScope is well-modularized toward flexibility and extendability for promoting the various FL applications.

### 3 FUNCTIONAL COMPONENTS

In this section, we introduce the functional components provided in FederatedScope for convenient usage, including personalization, auto-tuning, and privacy protection.

#### 3.1 Personalization

In an FL course, multiple participants aim to cooperatively learn a global model without directly sharing their private information. As a result, these participants can be arbitrarily different in terms of their underlying data distributions and system resources such as computational power and communication width. On the one hand, the differences among participants can lead to the data quantity skew, feature distribution skew, label distribution skew, and temporal skew, which are commonly observed in real-world applications. For example, the recorded videos by surveillance cameras located in different light conditions and filming angles can vary a lot. The standard FL course, which simply applies the shared global model for all participants, might hurt the model performance for some clients and lead to sub-optimal global performance. On the other hand, the active level and response time of participants can be diverse due to their different hardware capabilities and network conditions. Because of such systematical heterogeneity, it is challenging to make full use of local data in the standard FL course, and thus motivates Personalized Federated Learning (PFL) [15, 46].

3.1.1 **Built-in Algorithms.** The existing studies on personalized FL mostly propose to control the extent clients learn from the shared global model and how to fuse the shared knowledge with its local models. With the help of the proposed message-oriented framework, FederatedScope provides a flexible infrastructure to extend rich types of PFL algorithms, including: (i) Applying client-specific training configurations, e.g., hyperparameters; (ii) Integrating the global model with customized sub-modules; (iii) Customizing clients’ training behaviors, such as regularization and multi-model interaction; (iv) Changing server’s aggregating behaviors, such as model interpolation. Specifically, we implement fruitful state-of-the-art PFL algorithms, including pFedMe [45], FedBN [33], FedEM [37], and Ditto [31]. And we also provide the corresponding scripts to reproduce the reported results of these PFL algorithms.

#### 3.1.2 Evaluation.** Besides, existing studies on PFL adopt diverse evaluation metrics, diverse backbone models, and diverse datasets with different data sizes and non-IID settings. For example, the CIFAR-10/100 and FEMNIST datasets are widely adopted in [31, 37, 42, 45, 59], while with different simulation manners: [42] adopts the Dirichlet distribution based partition and [59] adopts the pathological partition for CIFAR-10/100; and the number of clients is 205 in [31] and 539 in [37] for FEMNIST. To conduct a fair comparison of the existing studies with consistent datasets and metrics, we establish more comprehensive evaluation methodologies and metrics in FederatedScope. The datasets and backbone models are unified in our DataZoo (in Section 4.1) and ModelZoo (in Section 4.2), while several evaluation metrics are also provided towards personalized FL, including global accuracy with diverse averaging manners, convergence monitoring and negative optimization monitoring for individual participants, accuracy standard deviation, and best or poorest accuracy deciles to indicate local model fairness.

#### 3.2 Auto-Tuning

A well-known fact is that the optimization of machine learning models is sensitive to hyperparameters, which becomes severe in FL applications partly due to the unstable convergence rooted from the non-IID issues. Furthermore, the tuning process can be extremely costly in federated learning, especially for the cross-device scenarios, since each trial needs more or fewer rounds of communication across participants. Therefore, besides the basic hyperparameter optimization (HPO) methods, we also include low-fidelity HPO strategy in FederatedScope.

3.2.1 **Build-in Methods.** We provide build-in methods aiming at alleviating the efforts of tuning hyperparameters for FL algorithms, including the basic HPO methods such as grid search, random search, gaussian process, etc. These general methods are applicable only to the simulation cases, where repeating a considerable number of FL courses is affordable. Besides, to balance effectiveness and efficiency in the tuning process, we provide low-fidelity HPO [34] strategy that achieves by executing just a few rounds or reducing the client sample rate. We have provided Hyperband [30] and PBT [28] to utilize the low-fidelity strategy, which enables HPO for FL algorithms in practice.

Meantime, we provide the implementation of the latest Federated HPO method FedEx [24], which is inspired by the weight-sharing strategy in neural architecture search and parameterize a policy to sample the hyperparameter configurations. Suppose there are $K$ possible configurations, then additional trainable parameters $a$, a $K$-dimensional real-valued vector, is considered to represent a stochastic policy. Specifically, we determine which configuration to use in each round by sampling from softmax$(a)$. It is worth noting that we sample for each considered client independently,
and thus the server broadcast both the latest model parameters and each client’s sampled configuration. Once a client received these messages, it re-specifies the configuration accordingly and conduct local updates as usual. Once the server received the performances from the considered clients, it makes aggregation for the model parameter as usual. Moreover, it needs to update \( \alpha \) based on these performances and their corresponding sampled actions (i.e., configurations). Since different clients might have different optimal hyperparameter configurations due to their non-IID data, we have implemented the personalization algorithms in a way that makes them also applicable to hyperparameters and encourage the community to conduct further exploration in this direction.

3.2.2 Customized. Here we present the design of the HPO module to help developers customize new HPO algorithms. In our implementation, we abstract the HPO method as an external agent which interplay with our FL runner in a trial-and-error manner. From such a point of view, the optimization procedure consists a series of attempts of hyperparameter configurations. Our FL runner has exposed the arguments for the agent to specify which configuration to be tried and which fidelity to be considered. Thus, all these mentioned HPO methods can be implemented in a unified way, and novel HPO method can be easily contributed to FederatedScope and compared to existing methods fairly.

In order to ease the manual tuning for customized algorithms, we have provided lots of metrics for local updates and local evaluations, including training/valid/test loss, accuracy, Hits@n, etc. More importantly, we have also provided several metrics, e.g., B-local dissimilarity [32], for our users to better monitoring the FL process, especially observing whether the convergence becomes difficult due to the large variance of collected gradients/parameters.

3.3 Privacy Protection

3.3.1 Differential Privacy. Differential privacy (DP) has always been a popular theoretical criterion for privacy protection, and has achieved great success in database [11, 13]. Specifically, a randomized mechanism satisfying \((\epsilon - \delta)\)-DP promises the privacy loss of all neighboring datasets is bounded by \(\epsilon\) with the probability of at least \(1 - \delta\) [14].

In recent studies [49, 53], DP techniques are applied in FL applications to enhance the privacy protection strength of exchange data, which injects artificial noise into shared parameters or gradients. In FederatedScope, we support DP algorithms by presetting various atomic modules as plugins, such as noise injection, weights/gradients clipping, etc. These modules can be opened/closed via configuring, and developers can combine different modules with suitable hyperparameters to implement a certain DP algorithm. As an example, we have implemented NbAFL [53] in FederatedScope, which provides privacy guarantees for both upload and download channels in federate learning. We conduct experiments for evaluation and show the detailed experimental results in Section 5.4.1.

3.3.2 Privacy attackers. The privacy attack algorithms are important and convenient to verify the privacy protection strength of the design of FL systems and algorithms, which is growing along with FL applications. However, most of the existing FL platform ignores including such an important functional component.

It is worth noting that the diversity of attack algorithms also brings challenges to the flexibility and extendability of an FL platform. According to the types of attacker’s target, typical attacks including membership inference attack, property inference attack, class representative attack and training data/label inference attack:

- In membership inference attack, a attacker can be a server or a client, and the objective is to infer whether the specific given data are exist in other clients’ private dataset;
- Property inference attack aims to infer s dataset property (may sensitive property) other than the class label. For example, in the facial image dataset, the original task is to infer whether wearing glass, a attacker may also curious about the gender and the age that is not related to original task;
- The goal of class representative attack is to infer the representative sample of specific class. This type of attack often exists in the case where a client only owns partial of class label, and he/she curious about the information related to other class;
- Training data/label inference attack aims to reconstruct the privately owned training samples through the intermediate information transmitted during an FL course.

Besides, according to the actions that the attacker made, the privacy attacks can be divided into passive attack and active attack. In passive attack, the attacker follows the FL protocols, and only saves the intermediate results or the received information for local attack computation. Due to the characteristics of passive attack, it is very hard to be detected by the FL monitor. Different from passive attack, in active attack, the attacker often injected malicious information into FL to induce other clients reveal more private information into the global model. The attacker preforming active attack are also named as malicious. Compared with passive attack, due to the malicious information injection, this type of attack is easier to be detected with additional information checking.

Benefiting from the message-oriented framework, FederatedScope supports developers to flexible customize malicious servers and clients that include attack behaviors. In order to make it easier for developers to verify the privacy protection strength, we provide implementations of state-of-the-art privacy attack algorithms in FederatedScope. Developers only need to wrap the original trainer with the implemented privacy attack trainers, which can be regarded as plugins and used in various FL applications. The implemented privacy algorithms including (i) Membership inference attack: Gradient inversion attack (active attack) [41]; (ii) property inference attack (passive attack) [40]; (iii) Class representative attack: DMU-GAN (active attack) [21]; (iv) training data/label inference attack (passive attack): DLG [62], iDLG [60], GRADINV [17].

4 BENCHMARK

One of the major purposes of FederatedScope is setting up an easy-to-use and comprehensive FL benchmark. To achieve this, we provide DataZoo, ModelZoo and AlgoZoo in FederatedScope for developers to conduct a fair and consistent comparison.
Table 1: The statistics of the datasets provided in DataZoo.

| Dataset       | Task                    | Num. of Instance | Subsample | Split Ratio | Num. of Clients |
|---------------|-------------------------|------------------|-----------|-------------|-----------------|
| FEMNIST       | Image Classification    | 805,263          | 5%        | 6:2:2       | 3,550           |
| CelebA        | Image Classification    | 200,288          | 10%       | 6:2:2       | 9,323           |
| Shakespeare   | Next Character Prediction | 4,226,158      | 20%       | 6:2:2       | 1,129           |
| Twitter       | Sentiment Analysis      | 1,600,498        | 5%        | 6:2:2       | 660,120         |
| Reddit        | Language Modeling       | 56,587,343       | 819 clients | 6:2:2       | 1,660,820       |
| DBLP (by venue)| Node Classification     | 52,202           | -         | 5:2:3       | 20              |
| DBLP (by publisher) | Node Classification | 52,202         | -         | 5:2:3       | 8               |
| Ciao          | Link Classification     | 565,300          | -         | 8:1:1       | 28              |
| MultiTask     | Graph Classification    | 18,661           | -         | 8:1:1       | 7               |

4.1 DataZoo

We collect and preprocess the widely-used datasets from various FL application scenarios, including computer vision datasets (FEMNIST [10], CelebA [35]) and natural language processing datasets (Shakespeare [38], Twitter [18], Reddit [39]) from LEAF [7], and graph learning datasets (DBLP [48], Ciao [47], MultiTask [55]) from EasyFGL. The statistics of these datasets are summarized in Table 1. Note that we follow the preprocess process from LEAF and EasyFGL for keeping the fair and consistent comparisons with existing studies on these datasets, including the partition mechanism (e.g., according to the writer of the digit/character for FEMNIST) and subsample ratio (e.g., only 5% of the total instances are used in the experiments for Shakespeare). And according to EasyFGL, FederatedScope provides various split methods for graph dataset, such as split by venue or by publisher.

Besides, we also provide a unified dataloader for the provided datasets in DataZoo, which allows developers to automatically download, preprocess, and fetch the data by configuring. Some changeable configurations, such as batch size and split ratio, can be determined by developers according to their computation resources and adopted algorithms.

4.2 ModelZoo

In FederatedScope, we provide ModelZoo to unify backbone models used in FL applications, which includes widely-used model architectures, such as ConvNet [27] and VGG [43] for computer vision tasks, LSTM [22] for natural language processing tasks, GNNs [25, 50, 56], GraphSAGE [19], GPR-GNN [9] for graph learning. With the help of ModelZoo, developers can conveniently build up various neural networks according to their FL applications via configuring.

Besides, FederatedScope allows developers to customize new model architectures or training configuration (e.g., optimizer) as in centralized training, since the federated behaviors and training behaviors are decoupled. To be more specific, a Trainer object is hosted by the each client to encapsulate the training details, and only exposes high-level interfaces such as TRAIN and EVAL to the client. The trainer serves as in a centralized training. More details can be found in Section 2.2.1. To search the suitable hyperparameters of models, developers can conduct HPO for both provided models and customized models via the provided auto-tuning modules (Section 3.2), and monitor the federal training process for further model tuning.

4.3 AlgoZoo

Various FL algorithms are provided in FederatedScope, such as FedOpt [2], FedNova [52], FedProx [32], etc. Compared to the vanilla FedAvg, these FL algorithms introduce new types of exchange messages and change the behaviors of servers and clients. For example, FedNova proposes to exchange the momentum among participants, and FedOpt utilizes an optimizer for aggregating rather than directly averages the received models.

The new types of messages and behaviors can be conveniently supported by the proposed message-oriented framework. In FederatedScope, we implement these FL algorithms as plugins, which can be used by specifying in the configurations. Note that most of the FL algorithms are orthogonal to the federal behaviors, model architectures, and training configurations, which means that developers can reuse these algorithms in different FL applications without repeatedly implementation.

Besides, FederatedScope integrates rich algorithms to handle the non-IID issues in FL applications. And we categorize these algorithms into PFL algorithms, which is provided in personalization component and introduce in Section 3.1.

5 EXPERIMENTS

We conduct a series of experiments with FederatedScope to evaluate its correctness and effectiveness, and provide the experimental results as references for the build-in models and algorithms. More experiments are running and more results will be reported soon.

5.1 Benchmarks

Firstly we report the experimental results for DataZoo, ModelZoo and AlgoZoo in Table 2. For each dataset, we chose the suggested model according to the literature, e.g., ConvNet2 for FEMNIST and GCN for Ciao. Besides, we adopt different FL algorithms for comparison, such as FedAvg, FedOpt, and FedProx. More experimental results will be reported later.

The experimental results imply that, different FL algorithms might be suitable for different FL applications. For example, FedProx achieves the best performance on Shakespeare, but is worse than
Table 2: The experimental results for DataZoo, ModelZoo and AlgoZoo.

| Dataset       | Model      | Algorithm | Accuracy (%) |
|---------------|------------|-----------|--------------|
| FEMNIST       | ConvNet2   | FedAvg    | 83.33        |
| FEMNIST       | ConvNet2   | FedOpt    | 84.92        |
| FEMNIST       | ConvNet2   | FedProx   | 84.77        |
| Shakespeare   | LSTM       | FedAvg    | 43.83        |
| Shakespeare   | LSTM       | FedOpt    | 47.39        |
| Shakespeare   | LSTM       | FedProx   | 47.85        |
| Reddit        | LSTM       | FedAvg    | 25.94        |
| DBLP (by venue)| GCN       | FedAvg    | 77.53        |
| DBLP (by publisher)| GCN | FedAvg | 77.98 |
| Ciao          | GCN        | FedAvg    | 49.18        |
| MultiTask     | GIN        | FedAvg    | 63.40        |
| MultiTask     | GIN        | FedOpt    | 63.33        |
| MultiTask     | GIN        | FedProx   | 63.01        |

v vanilla FedAvg on MultiTask. This phenomenon might be related to the type of the data, the heterogeneity of data distribution, the inductive bias of the adopted model, the number of sampled clients, and so on, which still remains a open question in federated learning.

5.2 Personalization

We provide empirical evaluation for build-in PFL algorithms, including FedBN, pFedMe, Ditto, and FedEM. The experimental results are shown in Table 3.

From the table we can observe that, PFL algorithms can bring significant improvements in some datasets, such as FEMNIST and MultiTask. However, the adopted personalized FL algorithms sometimes might hurt the model performance. For example, when applying pFedMe on Shakespeare, the model performance drop 6.43% compared to those without personalization. We empirically point out that the subsample of the datasets might bring randomness and instability in the experiments, thus developers might need a suitable configuration of hyperparameters to achieve a consistent boost in model performance.

Table 3: The results for build-in PFL algorithms.

| Dataset       | Personalized Algorithm | Accuracy (%) |
|---------------|------------------------|--------------|
| FEMNIST       | w/o                    | 83.33        |
| FEMNIST       | FedBN                  | 85.48 (↑ 2.15) |
| FEMNIST       | pFedMe                 | 87.65 (↑ 4.32) |
| FEMNIST       | Ditto                  | 86.61 (↑ 2.77) |
| FEMNIST       | FedEM                  | 84.79 (↑ 146) |
| Shakespeare   | w/o                    | 43.83        |
| Shakespeare   | pFedMe                 | 37.40 (↓ 6.43) |
| Shakespeare   | Ditto                  | 45.14 (↑ 1.31) |
| Shakespeare   | FedEM                  | 48.06 (↑ 4.23) |
| MultiTask     | w/o                    | 63.40        |
| MultiTask     | FedBN                  | 72.90 (↑ 9.50) |
| MultiTask     | ditto                  | 63.35 (↓ 0.05) |

5.3 Auto-Tuning

We evaluate the effectiveness of low-fidelity HPO provided in FederatedScope, with taking Successive Halving Algorithm (SHA) [30] and Random Search (RS) [3] as our HPO schedulers. We train a GCN model on the citation network Cora [4], and set the total training rounds to be 81 for SHA, and set the sample size to be 81, 27, and 9 with training round per trial as 1, 3, and 9, respectively.

5.4 Privacy Protection

We demonstrate the experimental results to evaluate the effectiveness of the provided privacy protection module, including build-in DP algorithms and implemented privacy attackers.

5.4.1 Differential Privacy. We train a ConvNet2 on FEMNIST with NbAFL [53] to achieve different (\(\epsilon, \theta\))-DP, and report the experimental results on Figure 6. We can observe that, with a smaller value of \(\epsilon\) and \(\theta\) (which indicates a larger privacy protection strength), the performance drop of the learned model is more significant, which demonstrates the shared data are less informative.

5.4.2 Privacy Attack Results. We conduct the DLG method [62] to perform training data inference attack for FedAvg on FEMNIST. In this attack setting, the server is the attacker, and it utilizes the optimization based method to search the data that can best match the received model update information that generated by the target clients. In this example, the total client number is 10, the target clients are all the clients, and FedAvg is set as single sample batch.
REFERENCES

[1] Manoj Ghishan Arivazhagan, Vinay Aggarwal, Asadiya Kumar Singh, and Sunav Choudhary. 2019. Federated learning with personalization layers. arXiv preprint arXiv:1912.00818 (2019).

[2] Muhammad Asad, Ahmed Moustafa, and Takayuki Ito. 2020. FedOpt: Towards Communication Efficiency and Privacy Preservation in Federated Learning. Applied Sciences 10, 8 (2020).

[3] James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. Journal of machine learning research 13, 2 (2012).

[4] Aleksandr Bojchevski and Stephan Günnemann. 2017. Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking. arXiv preprint arXiv:1707.03815 (2017).

[5] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dimitry Huia, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konečný, Stefano Mazocchi, Brendan McMahan, et al. 2019. Towards federated learning at scale: System design. Proceedings of Machine Learning and Systems 1 (2019), 374–388.

[6] Due Bui, Khuhit Malik, Jack Goetz, Honglei Liu, Seungwhan Moon, Anuj Kumar, and Kang G Shin. 2019. Federated user representation learning. arXiv preprint arXiv:1909.12535 (2019).

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

[8] Di Chai, Leye Wang, Kai Chen, and Qiang Yang. 2020. Fedeval: A benchmark system with a comprehensive evaluation model for federated learning. arXiv preprint arXiv:2011.09865 (2020).

[9] Eli Chien, Jianhao Peng, Fan Li, and Ogjice Milenkovic. 2021. Adaptive Universal Generalized PageRank Graph Neural Network. In International Conference on Learning Representations.

[10] Gregory Cohen, Saeed Aghazadeh, Jonathan Tapson, and Andre Van Schaik. 2017. EMNIST: Extending MNIST to handwritten letters. In 2017 international joint conference on neural networks (IJCNN). 2921–2926.

[11] Bolin Ding, Marianne Winslett, Jawei Han, and Zhenhui Li. 2011. Differentially private data cubes: optimizing noise sources and consistency. In Proceedings of the 2011 ACM SIGMOD International Conference on Management of data. 217–228.

[12] David A Duverle, Shohei Kawasaki, Yoshiti Yamada, Jun Sakuma, and Koji Tsuda. 2015. Privacy-preserving statistical analysis by exact logistic regression. In 2015 IEEE Security and Privacy Workshops. 7–16.

[13] Cynthia Dwork. 2008. Differential privacy: A survey of results. In International conference on theory and applications of models of computation. 1–19.

[14] Cynthia Dwork and Aaron Roth. 2014. The Algorithmic Foundations of Differential Privacy. Found. Trends Theor. Comput. Sci. 9, 3–4 (2014), 211–407.

[15] Alireza Fallah, Aryan Mokhtari, and Anasun Oztaklalar. 2020. Personalized federated learning: A meta-learning approach. arXiv preprint arXiv:2002.07984 (2020).

[16] Adria Gascon, Philipp Schoppmann, Borja Baile, Mariana Raykova, Jack Doerner, Sameer Zabur, and David Evans. 2017. Privacy-Preserving Distributed Linear Regression on High-Dimensional Data. Proc. Priv. Enhancing Technol. 2017, 4 (2017), 345–364.

[17] Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. 2017. Inverting gradients: how easy is it to break privacy in federated learning? Advances in Neural Information Processing Systems 30 (2020), 16937–16947.

[18] Alec Go, Richa Bhayani, and Lei Huang. 2009. Twitter sentiment classification using distant supervision. CS224N project report, Stanford 1, 12 (2009).

[19] Will Hamilton, Zhizhuo Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. Advances in neural information processing systems 30 (2017).

[20] Stephen Hardy, Wilko Henecka, Hamish Ivey-Law, Richard Nock, Giorgio Patrini, Guillaume Slama, and Brian Thorne. 2017. Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption. arXiv preprint arXiv:1711.10677 (2017).

[21] Briland Hitaj, Giuseppe Ateniese, and Fernando Perez-Cruz. 2017. Deep models under the GAN: Information leakage from collaborative deep learning. In Proceedings of the 2017 ACM SIGSAC conference on computer and communications security. 603–618.

[22] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. Neural computation 9, 8 (1997), 1735–1780.

[23] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Benis, Arjun Nitin Bhagoji, Kalista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. 2021. Advances and open problems in federated learning. Foundations and Trends® in Machine Learning 14, 1–2 (2021), 1–210.

[24] Mikhail Khodak, Renbo Tu, Tian Li, Liam Li, Maria-Florina Balcan, Virginia Smith, and Ameet Talwalkar. 2021. Federated hyperparameter tuning: Challenges, baselines, and connections to weight-sharing. Advances in Neural Information Processing Systems 34 (2021).

[25] Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In International Conference on Learning Representations.

[26] 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).

[27] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. nature 521, 7553 (2015), 436–444.

[28] Ang Li, Ola Spyras, Sagi Perel, Valentin Dalibard, Max Jaderberg, Chenjie Gu, David Budden, Tim Harley, and Pramod Gupta. 2019. A generalized framework for population based training. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 1791–1799.

[29] Li Li, Morning Duan, Duo Liu, Yu Zhang, Ao Ren, Xianzhang Chen, Yujian Tan, and Chengliang Wang. 2021. FedSAE: A novel self-adaptive federated learning framework in heterogeneous systems. In 2021 International Joint Conference on Neural Networks (IJCNN). 1–10.

[30] Lisha Li, Kevin G Jamieson, Afshin Rostamizadeh, Ameet Talwalkar. 2017. Hyperband: Bandit-based configuration evaluation for hyperparameter optimization. In ICLR (Poster).

[31] Xiaoxiao Li, Meirui Jiang, Xiaofei Zhang, Michael Kamp, and Qi Dou. 2021. Fedbn: Federated learning on non-iid features via local batch normalization. arXiv preprint arXiv:2102.07623 (2021).

6 CONCLUSIONS

In this paper, we introduce FederatedScope, a federated learning platform proposed for providing comprehensive and flexible supports for various FL applications. Towards both convenient usage and flexible extension, FederatedScope exploits a message-oriented framework to describe an FL course, which frames the FL course into multiple rounds of message passing among participants. In each round, the servers and clients exchange various types of messages and handle these messages accordingly. In this way, FederatedScope allows developers to customize different types of messages and the corresponding handling functions according to their FL applications. As a comprehensive FL platform, FederatedScope also includes rich functional components to promote frontier studies and real-world applications, such as personalization, auto-tuning, and privacy protection. Further, a benchmark consists of ModelZoo, DataZoo and AlgoZoo is provided in FederatedScope as a convenient tool for community to perform consistent and fair comparisons in various FL applications. We have released FederatedScope on https://github.com/alibaba/FederatedScope, with a detailed tutorial to help researchers and developers quickly get started, develop new FL algorithms, and build new FL applications.
[34] Yaliang Li, Zhen Wang, Yuexiang Xie, Bolin Ding, Kai Zeng, and Ce Zhang. 2021. Automl: From methodology to application. In Proceedings of the 30th ACM International Conference on Information & Knowledge Management. 4853–4856.

[35] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. 2015. Deep learning face attributes in the wild. In Proceedings of the IEEE international conference on computer vision. 3730–3738.

[36] Chuhan Wu, Fangzhao Wu, Yang Cao, Yongfeng Huang, and Xing Xie. 2021. FederatedScope: A Comprehensive and Flexible Federated Learning Platform via Message Passing.

[37] Othmane Marfoq, Giovanni Neglia, Aurélien Bellet, Laetitia Kameni, and Richard Vidal. 2021. Federated multi-task learning under a mixture of distributions. Advances in Neural Information Processing Systems 34 (2021).

[38] Brenda McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Brandon Bacciu. 2019. FedNets: Federated neural networks. In Advances in Neural Information Processing Systems 32 (2019).

[39] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How Powerful are Graph Neural Networks? In International Conference on Learning Representations.

[40] Canh T Dinh, Nguyen Tran, and Josh Nguyen. 2020. Personalized federated learning with moreau envelopes. Advances in Neural Information Processing Systems 33 (2020). 21394–21405.

[41] Alysa Ziying Tan, Han Yu, Lizhen Cui, and Qiang Yang. 2021. Towards personalized federated learning. arXiv preprint arXiv:2105.00740 (2021).

[42] Zihao Wang, Zikhaiy Kaplin, Di Ni, and Bochun Li. 2020. Optimizing federated learning on non-iid data with reinforcement learning. In IEEE INFOCOM 2020—IEEE Conference on Computer Communications. 1698–1707.

[43] Yi Zhou, Parikshit Ram, Theodoros Salonidis, Nathalie Baracaldo, Horst Samulowitz, and Heiko Ludwig. 2021. FLoRA: Single-shot Hyper-parameter Optimization for Federated Learning. arXiv preprint arXiv:2112.08524 (2021).

[44] Liqeng Zhu, Zhijian Liu, and Song Han. 2019. Deep leakage from gradients. Advances in Neural Information Processing Systems 32 (2019).

[45] Alexander Ziller, Andrew Trask, Antonio Lopardo, Benjamin Szymkow, Bobby Wagner, Emma Bluemek, Jean-Mickael Nounahon, Jonathan Passerat-Palmbach, Kritika Prakash, Nick Rose, et al. 2021. Pysyft: A library for easy federated learning. In Federated Learning Systems 111–139.