Felicitas: Federated Learning in Distributed Cross Device Collaborative Frameworks

Qi Zhang
zhangqi92@huawei.com
Central Software Institute, Huawei Technologies Co. Ltd
China

Tiancheng Wu*
wutiancheng@huawei.com
Central Software Institute, Huawei Technologies Co. Ltd
China

Peichen Zhou
zhoupeichen@huawei.com
Central Software Institute, Huawei Technologies Co. Ltd
China

Shan Zhou
zhoushan7@huawei.com
Central Software Institute, Huawei Technologies Co. Ltd
China

Yuan Yang
yangyuan24@huawei.com
Central Software Institute, Huawei Technologies Co. Ltd
China

Xiulang Jin
jinxiulang@huawei.com
Central Software Institute, Huawei Technologies Co. Ltd
China

ABSTRACT

Felicitas is a distributed cross-device Federated Learning (FL) framework to solve the industrial difficulties of FL in large-scale device deployment scenarios. In Felicitas, FL-Clients are deployed on mobile or embedded devices, while FL-Server is deployed on the cloud platform. We also summarize the challenges of FL deployment in industrial cross-device scenarios (massively parallel, stateless clients, non-use of client identifiers, highly unreliable, unsteady and complex deployment), and provide reliable solutions. We provide the source code and documents at https://www.mindspore.cn/. In addition, the Felicitas has been deployed on mobile phones in real world. At the end of the paper, we demonstrate the validity of the framework through experiments.

CCS CONCEPTS

• **Computer systems organization → Client-server architectures**; **Dependable and fault-tolerant systems and networks**.

KEYWORDS

cross-device federated learning, distributed framework, large-scale, data mining under privacy constraints

ACM Reference Format:
Qi Zhang, Tiancheng Wu, Peichen Zhou, Shan Zhou, Yuan Yang, and Xiulang Jin. 2022. Felicitas: Federated Learning in Distributed Cross Device Collaborative Frameworks. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '22), August 14–18, 2022, Washington, DC, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3534678.3539039

*corresponding author

KDD '22, August 14–18, 2022, Washington, DC, USA.

ACM ISBN 978-1-4503-9385-0/22/08…$15.00
https://doi.org/10.1145/3534678.3539039

1 INTRODUCTION

Mobile phones, wearable devices and other Internet of Things devices, generate a wealth of data everyday. Due to the growing computational power of these devices, coupled with their privacy concerns and communication cost, it is increasingly attractive to store data locally and push network computations to the edge devices. With Federated Learning (FL) [1], each FL-Client device can attain a global view and predict a situation that is seen in another device and ultimately achieves usable generalization capability in the wild. The currently widely accepted classifications of FL settings are "cross-device" and "cross-silo" [2]. They vary greatly in distribution scale, data availability, FL-Client reliability, etc.

The growing demand for FL technology has resulted in a number of tools and frameworks becoming available. These include TensorFlow Federated (TFF) [3], which provides two different levels of APIs for developers to declaratively express federated computations. Federated AI Technology Enabler (FATE) [4] is an open source industry-level FL framework proposed by the WeBank AI Department. The FATE implements a secure computing protocol based on homomorphic encryption and multi-party computing. It supports a series of FL related algorithms, including logistic regression, decision tree, gradient boost tree, deep learning, and transfer learning. Multiple companies and organizations can cooperate to train machine learning models while ensuring data security. LEAF [5] is a federated learning reference framework released by CMU (Google was involved in writing the paper), including multi-tasking, meta-learning, device-side learning, and various datasets. PySyft [6] is a security and privacy deep learning Python library released by UCL, DeepMind, and OpenMind, including federated learning, differential privacy, and multi-party learning. PySyft uses PyTorch and Tensorflow as the training framework. In addition, commercial data platforms incorporating FL are still in development stage from established technology companies as well as smaller start-ups: PaddleFL [7] and Clara [8].

However, most of the existing FL frameworks are only suitable for experimental scenarios. In addition, these frameworks cannot be directly applied in cross-device FL scenarios. We summarize the challenges of cross-device FL in industrial scenarios as follows:

- FL-Clients cannot be indexed directly and long links between FL-Clients and FL-Server cannot be maintained (due to the restriction on resources).
- It is not easy to deploy models across multiple embedded hardware architectures.
- There are millions of FL-Clients to be processed in parallel.
- Cross-device FL iterations will have a long-tail effect caused by untimely FL-Client response or even falling behind.
- Privacy and security challenges: How to ensure that the FL-Clients and FL-Server are not malicious to protect privacy and guarantee security.
- Lacking labeled data is a common problem at device-side.

To overcome these challenges, in this paper, we introduce Felicitas, a distributed end-to-end cross-device FL framework:

- To address the challenges that devices with limited resources cannot be directly indexed and maintain persistent connection. In our framework, the communication between the FL-Clients and the FL-Server is initiated by the FL-Clients, and then the FL-Server functions as a service-oriented distributed cluster through the Elastic Load Balance.
- To run millions device in parallel, we propose a new distributed paradigm of aggregate computing. Several FL-Servers can gather data from random FL-Clients.
- To ensure deployment of heterogeneous endpoints, our FL-Clients are built on the device-side deep learning framework and maintains a unified intermediate representation (IR) and a unified runtime with the cloud. The cloud-based training model can be seamlessly deployed on the device side.
- In an FL system, there will always be a small number of responses with higher-than-average delays, and this phenomenon is called the "long tail effect". To address the "long tail effect" caused by heterogeneous devices, we implement a FL-Server framework with distributed servitization and consistent computing, and set up a time-limited communication module.
- To protect privacy and security, we implement model encryption algorithms based on multi-party security computing with lossless precision and a local differential privacy algorithm with high efficiency.
- To address the lacking labeled data challenge, Felicitas provides a hybrid training mode and a new role of FL-Worker on FL-Server. The FL-Server not only implements federated aggregation, but also conducts supervised training.

In addition, Felicitas encapsulates the FL process in the C++ execution layer of the deep learning framework. Python primitives are used to define algorithms and FL-plans to improve aggregation computing efficiency and reduce learning costs. Users can train deep neural network on data which never leaves their devices. Last but not the least, we verfiy the effectivenss and efficeny of our framework with extensive experiments.

2 ARCHITECTURE

The Felicitas architecture, as shown in Figure 1, involves two parts: FL-Server maintains partitions for global shared parameters (machine local parameters are not synchronized by default). They communicate regularly to calculate parameters for reliability and scalability. The FL-Clients are used to calculate local statistics such as gradients or parameters. FL-Clients only communicate with the FL-Server. The FL-Clients can decide when to join or quit by themselves, and the FL-Server has no control over the FL-Clients. FL-Server mainly performs bookkeeping and global aggregation steps with data uploaded by the FL-Clients.

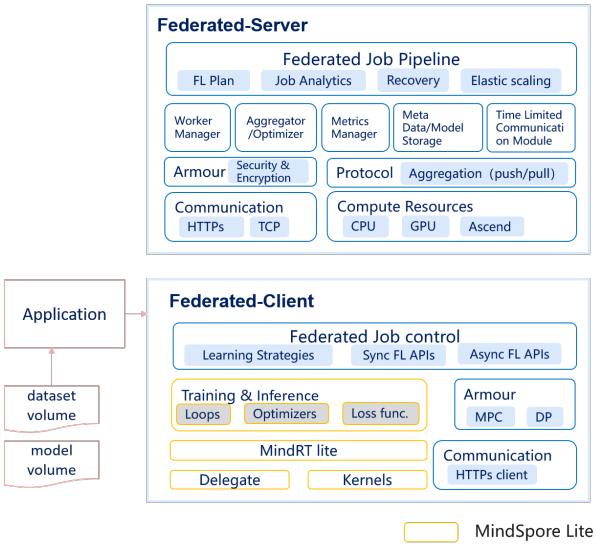

Figure 1: Felicitas architecture

The core design of the Felicitas is to consider how to make the distributed FL process work efficiently and collaboratively. Specifically, compared to the existing open-sourced FL framework, we have implemented the following unique features:

1) Unlike the existing ones that separate communication computing and federated aggregation from the deep learning framework, which make users to express the interaction process of FL completely, Felicitas allows switching between standalone and federated training model with one line of code, a simple and friendly implementation mode. For details, see Section 2.1.
2) Unlike ordinary distributed training and existing FL frameworks, we do not consider FL-Clients as part of a stable network that maintains long links with FL-Server. Section 2.2 describes the communication links and networking modes provided by the Felicitas to support the deployment on millions of devices, which can tolerate delays and exits of unstable FL-Clients at any moment.
3) Design based on the above two basics, we have implemented a novel distributed algorithm paradigm of federated aggregation, in Section 2.3. Based on this paradigm, we build service-oriented elastic scaling and fault tolerance capabilities, enhancing the serviceability and robustness of the overall framework, see Section 2.4.
4) To protect data privacy and defend robustness attack, we realized local differential privacy (LDP), secure multi-party computation (SMPC) and combined a client-server authentication protocol with a random FL-Client selection policy, see Section 2.5.

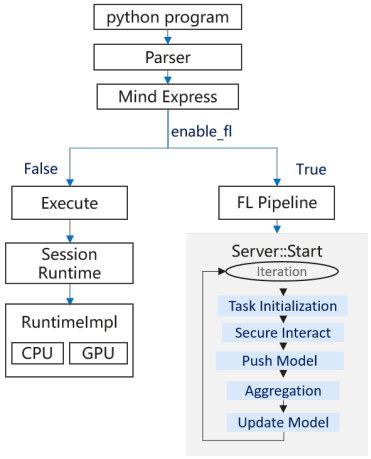

Figure 2: Felicitas execution process

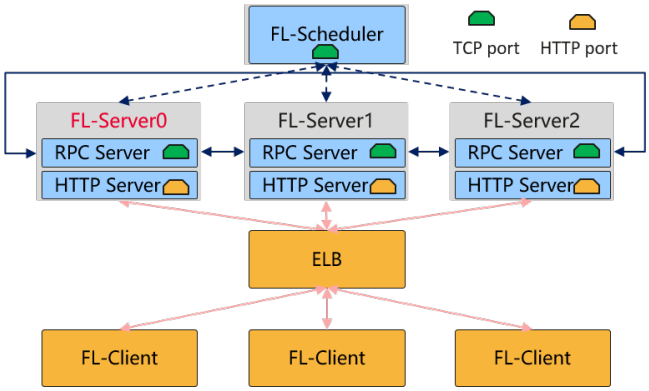

Figure 3: Physical architecture diagram of FL-Server networking

5) As shown in Section 2.6, we have developed a novel a semi-supervised hybrid training mode to extend FL deployment applications to end-side with few labeled data.

2.1 Interface

Felicitas has two phases: CompileTime and RunTime. In CompileTime, user-defined neural networks and distributed environment information are compiled into a static graph execution plan. Plan consists of the description information of the execution unit kernel. In RunTime, a FL plan is executed on FL-Server and FL-Clients in distributed clusters, and the basic execution unit is kernel. The data generated and consumed by the kernels is stored in Metastorage. The kernels run cooperatively through FL-Client message transfer and some status mechanisms.

The algorithm definition method of the FL script is the same as that of the traditional data center training method, which reduces the cost of learning the framework and facilitates the scalability of the framework. Here, we offer an example of using Felicitas.

```
import felicitas.context as context
ctx = {
    "enable_fl": True,
    "server_mode": "FEDERATED_LEARNING",
    "ms_role": "MS_SERVER",
    "worker_num": 0,
    "server_num": 1,
    "scheduler_ip": "127.0.0.1",
    "scheduler_port": 8113,
    "fl_server_port": 6666,
    "start_fl_job_cnt": 1,
    "fl_name": "Lenet",
    "iteration_num": 200,
    "epoch_num": 300,
    "batch_size": 32,
}
context.set_context(mode=context.GRAPH_MODE)
context.set_fl_context(**ctx)
```

As this example shows, there are only a few changes necessary to make single-machine program to cross-device FL program:

- Hyper-parameters required for FL are transferred through Python command line parameters.
- Transfer the hyper-parameter to context and set "enable_fl" to True.
- Define the algorithm network and training process in the same way as the single-machine training method.

The preceding process shows that the definition of FL jobs in Felicitas is similar to that in common deep learning training scripts except that some hyper-parameters are added. The main reason is that the Python front-end expression and graph compilation processes in the deep learning framework are reused, as shown in the Figure 2.

When "enable_fl" in the context is disabled, the framework follows the deep learning training process of graph optimization and graph execution by default. When "enable_fl" is enabled, the framework switches to the FL pipeline, and the listening port of the server is enabled to wait for different message requests from the FL-Client.

2.2 Network Frame

In Felicitas, FL-Server is a resident service deployed on the cloud platform and waits for information from FL-Clients to arrive. FL-Server has three roles: FL-Scheduler, FL leader server node, and FL follower server node. The FL-Scheduler maintains a networking structure. The FL leader server node has a state machine and maintains a global time-limited communication apparatus in each device-cloud interaction as a trigger point for message flow or failure of the follower server node. The FL follower server node is stateless and consists of a series of kernel rounds and is driven by different messages on the device or manager server node. Figure 3 shows the structure of the entire network.

The characteristics and advantages of this networking are as follows: FL-Clients access FL services in loose coupling mode, and FL-Server interacts with FL-Clients in request-response mode. There is consistency guarantee in FL-Server and aggregation parameters are calculated by federated. Then FL-Clients can obtain correct parameter information when requesting any server. From the perspective of FL-Clients, there is no need to maintain a stable connection between FL-Clients and FL-Servers. All FL-Clients only need to access the FL-Server cluster with unique addresses. The cluster routes FL-Clients requests to each FL-Server node based on the load balancing policy. The FL-Clients can exit freely.

This networking also brings out the performance challenge of "long tail effect" caused by the variation on storage capacity, computing power and communication capability between heterogeneous devices. To address this challenge, FL-Server implements the time-limited communication device, as shown in Figure 4.

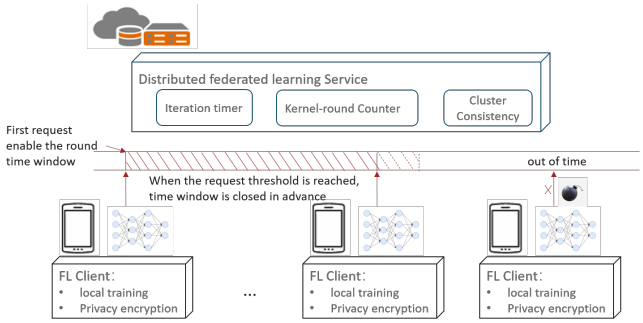

Figure 4: Time-limited communication device

This device is triggered by the first message received by the server cluster in each round of iteration, and only messages within the time window are processed in each round of interaction, otherwise simply ignored. In this way, the long tail effect caused by the FL-Client's untimely response is solved. Our time-limited communication device can also close the flow in advance, which further improves the efficiency of FL when the number of FL-Clients in the current round of interaction can reach the threshold ahead of time.

2.3 Distributed Federated Aggregation

In the face of a large influx of FL-Client data, the solution of Google is hierarchical aggregation [3]. A master aggregator manages the rounds of each FL task and generates several sub-aggregators based on the number of devices and update size, and then the master aggregator completes the final aggregation. In this manner, the primary aggregator becomes a performance bottleneck of communication and computing, and disaster recovery cannot be performed for the primary aggregator.

We propose a new distributed federated aggregation paradigm, as shown in Figure 5. Several FL-Servers gather data from random FL-Clients. Based on the network architecture designed in Section 2.2, FL-Scheduler is only responsible for maintaining the

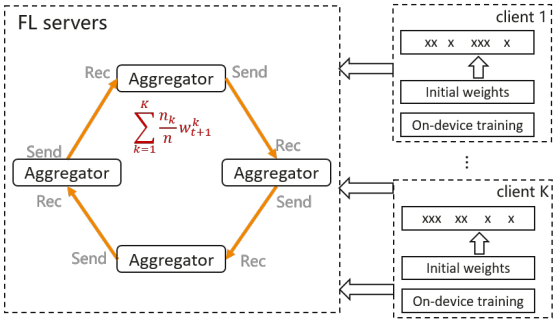

Figure 5: distributed federated aggregation

Algorithm 1 Distributed Aggregation for FedAvg

Input: FL-Server set S
for $t = 0$ **to** $T - 1$ **do**
  Randomly generate subset C of FL-Clients
  // Running on FL-Clients
  for FL-Client i **in** C in parallel **do**
    Randomly select a server j from S
    Receive parameter sp^j from server j
    load_parameter_into_model(model, sp^j)
    train(model)
    cp_i^j = get_parameter(model)
    Randomly select a server k from S
    Upload parameter cp_i^j and data size cs_i^j from FL-Client i to server k
  end for
  // Running on FL-Servers
  for server j **in** S in parallel **do**
    Receive parameter set cp^j and data size set cs^j from FL-Clients
    $sp^j = \sum_{i \in cp^j} cs_i^j * cp_i^j$
    $ss^j = \sum_{i \in cs^j} cs_i^j$
    $sp, ss = ring\_allreduce(sp^j, ss^j)$
    Wait for all FL-Servers have done $ring\_allreduce$
    $sp = sp/ss$
    Download parameter sp^j from server j to FL-Clients
  end for
end for

networking, but does not participate in actual services and computing. FL-Servers know the existence of each other and rank ID through heartbeats with FL-Scheduler. Then, a logical ring is formed between all server processes. Each process communicates with only two adjacent processes in the logical ring topology. The advantage of this method is that communication and computing overheads of the original primary aggregator are dispersed, and each aggregator can obtain a correct global FL-Client aggregation result after the Ring-AllReduce [9] process. Each aggregator is equal, so the system can easily implement disaster recovery and scaling.

The distributed federated aggregation method can be integrated into any federated aggregation algorithm. To make this method easier to understand, we use the FedAvg algorithm as an example. As shown in Algorithm 1, in the tth iteration, each FL-Client uploads the trained parameters and data size to a randomly selected server from the FL-Server set. After all jobs on FL-Clients have done, jobs on FL-Servers begin. First, each server locally aggregates the parameters by both the received parameters and data sizes. Second, the Ring-AllReduce is used to obtain the global aggregation results of all FL-Servers. Third, the parameters can be downloaded only after the Ring-AllReduce processes have done on all FL-Servers.

2.4 Elastic Scalability and Fault Tolerance

In a device-cloud commercial environment, FL-Scheduler and FL-Server are usually deployed on Elastic cloud Servers (ECSs) or Cloud Container Engines (CCEs). The cluster manager of the cloud platform can apply for and release these resources at any time.

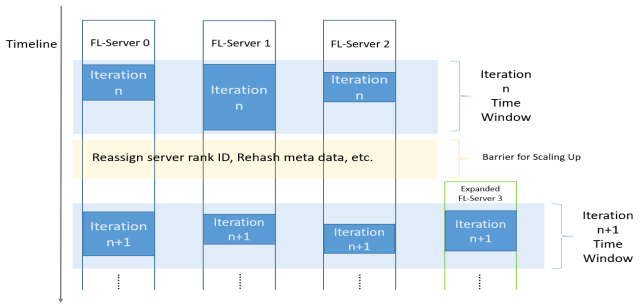

Figure 6: The timeline of scaling up one FL-Server

Device-cloud FL is usually performed when mobile phones are idle at night. To fully utilize computing resources on the cloud platform and reduce costs, FL-Server nodes should be released in their spare time and expanded during training. Therefore, elastic scaling and fault tolerance of the framework are required.

But in the most FL frameworks, neither elastic scalability nor fault tolerance is not well supported. Once migrated to a production environment, these frameworks will not be robust due to the instability of hardware system resources and the complexity of the network environment, which will lead to termination of training jobs and even data loss. This is not acceptable in a production environment.

For the framework of resident services, the timing of exceptions is unpredictable, and the commands that trigger scale-in/scale-out and the timing of node faults often occur during the iteration of the learning job, for example, when the FL-Server is performing joint computation. A common solution is to abandon this iteration and continue, but the remaining state of this iteration may also affect the next iteration, making it difficult to maintain correctness and metadata consistency of the joint computation. This poses a major challenge for Felicitas: how to ensure the correctness of the training results while keeping the training process running as it scales up and down. Felicitas offers the following solution:

- Scaling timing: Felicitas ingeniously implements the time-limited communication device described in Section 2.2. Since the time cost of each iteration of the training process is limited to a specific value, the distributed federated aggregation, FL rounds, model storage and evaluation are completed in this time window as shown in Figure 6. This gives Felicitas a chance to prepare for scaling up and down FL-Server and reset parameters and training metadata only after one iteration is completed.
- Data recovery: To support distributed storage for a large load of metadata in FL-Server, like model, public/private keys, device information, etc, Felicitas applies consistent hash algorithm [10]. FL-Servers and metadata are mapped onto a hash ring so that only a small amount of data needs to be migrated whenever scaling up/down happens. In addition, these data is also written into database. So in fault tolerance scenarios, if one FL-Server is down, the metadata on this server can be migrated to next FL-Server on the ring clockwise. So the training process can proceed with all metadata available to FL-Servers.

2.5 Privacy and Robustness

Existing cross device collaborative frameworks might be vulnerable to various privacy and robustness attacks.

- **Privacy**: Communicating plain-text model parameters can reveal sensitive information of FL-Clients' local data.
- **Robustness**: Malicious FL-Clients can poison the global model by attacking the convergence of the global model or injecting backdoor into the global model.

Felicitas provides two privacy protection and robustness enhancement algorithms based on local differential privacy (LDP) [11] and secure multi-party computation (SMPC) [12].

If the LDP method is used, each FL-Client first performs per-dimension or l_2-norm clipping on the parameter matrix W of local model, and then generates a differential noise (e.g., Gaussin noise) matrix G accoding to the privacy budget ϵ. The generated noise G is added to the clipped parameter matrix W' to obtain the parameter matrix W_p that meets the differential privacy definition and prevents the privacy attacks:

$$W_p = W' + G \tag{1}$$

The FL-Client uploads W_p to the FL-Server for federated aggregation. According to the law of large numbers, the aggregated global model can cancel most of the noise when the number of participants is large enough. The LDP-based aggregation solution incurs negligible perfomance overhead but can decrease the accuracy of global model when the number of participants is small.

To protect privacy with a small number of participants and without compromising accuracy, Felicitas also provides the SMPC-based method. In this method, any two FL-Clients u and v first aggree on a pair of random perturbations p_{uv} and p_{vu}:

$$p_{uv} = \begin{cases} -p_{vu}, & u \neq v \\ 0, & u = v \end{cases} \tag{2}$$

Then, the perturbations $p_{uv}, v \in U$ are added to the original model parameters x_u to obtain the perturbed model y_u:

$$y_u = x_u + \sum_{v \in U} p_{uv} \tag{3}$$

Finally, the FL-Server recevies the perturbed model y_u and computes the aggregation result $\overline{x}$ is as follows:

$$\begin{aligned} \overline{x} &= \sum_{u \in U} (x_u + \sum_{v \in U} p_{uv}) \\ &= \sum_{u \in U} x_u + \sum_{u \in U} \sum_{v \in U} p_{uv} \\ &= \sum_{u \in U} x_u \end{aligned} \tag{4}$$

The preceding process describes only the main idea of the aggregation algorithm. The SMPC-based aggregation solution is accuracy-lossless but increases the number of communication rounds.

To prevent robustness attacks, Felicitas combines a client-server authentication protocol with a random FL-Client selection policy. Specifically, in the FL-Client selection stage, a device which requests to join a FL job should also send its authentication information (e.g., certificates and digital signatures) to the central server. The central

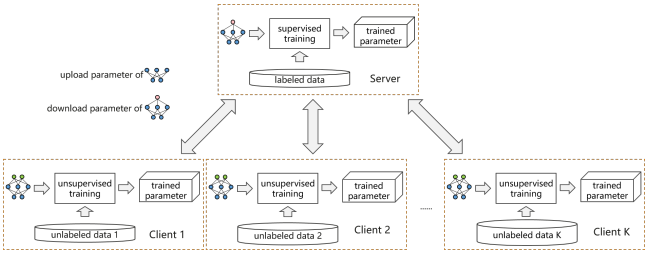

Figure 7: Semi-supervised Federal Hybrid Training Flowchart

Algorithm 2 Semi-supervised Federal Hybrid Training

initialization
for $t = 0$ **to** $T - 1$ **do**
  Randomly generate subset C of FL-Clients
  // Running on FL-Clients
  for FL-Client i **in** C in parallel **do**
    Receive parameter sp from FL-Server
    load_parameter_into_model($\text{model}_{\text{unsupervised}}$, sp)
    unsupervised_train($\text{model}_{\text{unsupervised}}$)
    cp_i = get_parameter($\text{model}_{\text{unsupervised}}$)
    Upload parameter cp_i and data size cs_i from FL-Client i to FL-Server
  end for
  // Running on FL-Server
  Receive parameter set cp and data size set cs from FL-Clients
  $sp = \sum_{i \in C} cs_i * cp_i$
  $ss = \sum_{i \in C} cs_i$
  $sp = \frac{sp}{ss}$
  Transfer parameter sp from FL-Server to FL-Worker
  // Running on FL-Worker
  Receive parameter sp from FL-Server
  load_parameter_into_model($\text{model}_{\text{supervised}}$, sp)
  supervised_train($\text{model}_{\text{supervised}}$)
  sp = get_parameter($\text{model}_{\text{supervised}}$)
  Transfer parameter sp from FL-Worker to FL-Server
  // Running on FL-Server
  Receive parameter sp from FL-Worker
  Download parameter sp from FL-Server to FL-Clients
end for

server will check the legitimacy of the requests and randomly select a small partial of devices with legitimate request to join the FL job. A request is considered as legitimate only if the corresponding device has valid authentication information and the device has not been added into the FL job in current iteration.

2.6 Semi-supervised Federal Hybrid Training

In the standard FL scenario, labeled data involved in model training is stored on the FL-Client. That is, standard supervised training is performed on the FL-Client. However, in an actual application scenario, labeled data is expensive, and a large amount of unlabeled data is stored on FL-Clients.

In the Felicitas, a semi-supervised horizontal FL mode is proposed. We add the FL-Worker role, which is used for supervised training on the server side. It is worth mentioning that the FL-Worker role on the server is necessary considering the cost of actual deployment. Because, if the FL-Server is directly responsible for expensive training tasks, multiple expensive machines are required to deploy the FL-Server cluster.

As shown in Figure 7, in the FL process, the FL-Worker's supervised training is alternately performed with the FL-Clients' unsupervised training. Except for the output layer, the feature extraction model structure of the subject is the same. Between alternating intervals, the both models only load the received parameters of the intersection into themselves. Algorithm 2 shows the pseudo-code of semi-supervised hybrid training.

3 EXPERIMENTS

Experiments are divided into four sections to verify: (1) Felicitas is an effective FL framework; (2) The proposed framework is suited to large-scale FL; (3) Felicitas can implement parameter encryption without loss of accuracy; (4) In real-world scenarios, Felicitas can use unlabeled data to improve model accuracy.

3.1 Comparison with Existing Frameworks

In this section, we compare Felicitas with other frameworks on common datasets in the field of FL. All frameworks use FedAvg to aggregate parameters. In FEMNIST [5], CIFAR-100 [13], and Shakespeare [1], 10 FL-Clients are randomly selected in each iteration for training. In StackOverflow [14], 50 FL-Clients are randomly selected from each iteration. For fair comparison, the datasets and models are the same across different frameworks. Further details on datasets, models, and experimental settings can be found in FedML [15] and FedOPT [16]. Table 1 shows the accuracy of the different models on different frameworks and datasets. As can be seen, the results for all frameworks are similar, as we employ the same algorithms and hyper-parameters. This set of experiments proves that under the same configuration, our framework does not differ significantly from other frameworks in accuracy.

3.2 Impact of Large-Scale FL-Clients

The experiment in this subsection is based on LeNET. The FL-Server is deployed on a Ubuntu x86_64 server with 12 CPU cores and 32 GB memory. The FL-Client uses the Python 3.7 requests library to send requests to simulate real mobile phone behavior. The parameter size of the LeNET model is about 266085 bytes.

Experimental results are displayed on Table 2. The average waiting time for FL-Clients to update and get weights for multiple iterations is used as the indicator. Apparently, with the distributed federated aggregation method of Felicitas, the time of communication can be greatly reduced by increasing the number of servers. Felicitas greatly reduces the high communication cost when deploying large-scale mobile devices.

3.3 Impact of Security and Privacy Method

In the conventional FL framework, models are uploaded to the central server in plain-text. If an attacker steals a plain-text model, the users' personal data could be restored through attacks such as reconstruction and model inversion. Felicitas uses the SMPC algorithm to encrypt parameters to prevent privacy leakage. This

Table 1: Accuracy on Different Datasets

Dataset	Model	Felicitas	TFF	FedML	LEAF
FEMNIST	LeNET	79.4	84.8	80.2	74.7
CIFAR100	ResNet18	37.3	39.1	34.0	-
Shakespeare	LSTM	56.5	57.0	57.1	52.1
StackOverflow	LSTM	23.4	15.6	18.3	-

Table 2: Average Get Model Wait Time

Server Number	Client Number	Wait Time(ms)
1	5000	5112.517
1	10000	21671.33
4	5000	456.4
4	10000	3698.07

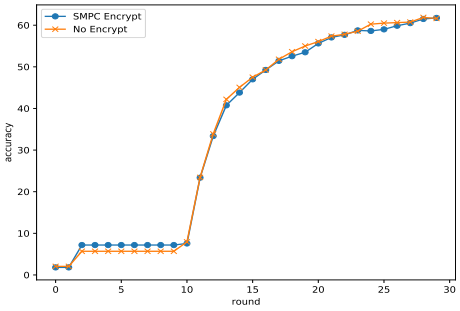

Figure 8: Accuracy of Algorithms with and without SMPC

Table 3: Hyper-parameter Settings

Hyper-parameters	Settings
Sequence Length on Unsupervised Model	16
Sequence Length on Supervised Model	128
Vocabulary Size	11682
Batch Size	16
Learning Rate	1e-5
Iteration	10
Client Numbers Per Iteration	30
Epoch Numbers Per Iteration	1
Loss Function	Cross Entropy
Optimizer	AdamW

Table 4: Accuracy on Different Training Methods

Methods	Accuracy
Semi-Supervised Federated Training	84.6 (±1.1)
Supervised Training	81.4 (±1.2)

section uses FEMNIST as experimental dataset. Most of the experimental configurations are the same as in Section 3.1, except for 500 FL-Clients are randomly selected, and 30 iterations are used for training. In fact, any number of FL-Clients and iterations has no effect on the conclusion of the experiment, and we use those numbers to reduce the experimental training cost. As shown in Figure 8, the accuracy is almost the same in the same iteration with or without encryption algorithms.

3.4 Experiments in Real-World Scenarios

In the actual scenario, we apply Felicitas to user profile construction. In China, the application has been deployed on mobile phones with the latest Huawei Mobile Services. You can view it under Settings > Privacy > Advertising and Privacy > More Information. Specifically, we classified each SMS message on each mobile phone into 5 classes and used the classification result as the user profile of the mobile phone owner. In order to give the simulation results, we collected 800,000 real SMS messages on the premise of informing users, and labeled 1,500 of them. The unlabeled data was randomly divided into 300 parts to each FL-Client, and each contains between 16 and 50,000 SMS messages. The labeled data was randomly divided into training sets and test sets in a ratio of 1:2.

We use a tiny version of the ALBERT algorithm [17]. The model used for unsupervised training uses the mask language model in BERT [18] as the decoder. Unsupervised training tasks are the commonly used mask language task in natural language processing. The model for supervised training uses a fully-connected network as the output layer. For more details of the model and training hyper-parameters, refer to Table 3.

We performed experiments with different random seeds. Table 4 shows the mean (maximum volatility) accuracies of supervised training only on single machine and semi-supervised federated training on both FL-Clients and FL-Server. Semi-supervised method leads to a 3.2-point improvement. The reason why the accuracy can be improved is that the unsupervised task uses unlabeled data to improve the feature extraction capability of the model, and the model in supervised task can use this feature extraction part to improve the performance of the model on downstream task.

4 CONCLUSION AND FUTURE WORK

We summarize the difficulties of deploying FL frameworks in the real world, and propose a distributed cross-device and industry-oriented FL framework which can solve them. The Felicitas is opened source on MindSpore[1] and has been deployed in real-world scenarios. The experimental results we present also demonstrate the validity of our framework. In the future, we will focus on:

[1]Documents can be referred to: https://www.mindspore.cn/federated/docs/zh-CN/master/index.html

- Integrating more FL related algorithms into Felicitas, such as federated optimization algorithms [19][20][21], compression algorithms [22][23][24] and privacy protection algorithms [25][26][27].
- Provides interfaces for uploading more statistics to facilitate user tracing algorithm performance or personalized deployment of FL algorithms.
- Design benchmarks, operators and baselines with high cohesion and low coupling. Academic or industrial users do not need to implement a FL framework from the ground up, but only need to modify the code of the modules in different parts to achieve algorithm innovation and improvement.
- How could the Felicitas be compatible with the recently developed clustered FL [28][29][30] and personalized FL [31][32][33] is a problem we will be solving in the future.
- FL for heterogeneous clients, especially the heterogeneous model architecture [34][35][36], will also be our future research direction.

REFERENCES

[1] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, 54:1273–1282, 2017.

[2] Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, Rafael G. L. D'Oliveira, Hubert Eichner, Salim El Rouayheb, David Evans, Josh Gardner, Zachary Garrett, Adrià Gascón, Badih Ghazi, Phillip B. Gibbons, Marco Gruteser, Zaid Harchaoui, Chaoyang He, Lie He, Zhouyuan Huo, Ben Hutchinson, Justin Hsu, Martin Jaggi, Tara Javidi, Gauri Joshi, Mikhail Khodak, Jakub Konecný, Aleksandra Korolova, Farinaz Koushanfar, Sanmi Koyejo, Tancrède Lepoint, Yang Liu, Prateek Mittal, Mehryar Mohri, Richard Nock, Ayfer Özgür, Rasmus Pagh, Hang Qi, Daniel Ramage, Ramesh Raskar, Mariana Raykova, Dawn Song, Weikang Song, Sebastian U. Stich, Ziteng Sun, Ananda Theertha Suresh, Florian Tramèr, Praneeth Vepakomma, Jianyu Wang, Li Xiong, Zheng Xu, Qiang Yang, Felix X. Yu, Han Yu, and Sen Zhao. Advances and Open Problems in Federated Learning. *Foundations and Trends® in Machine Learning*. 2021.

[3] Alex Ingerman and Krzys Ostrowski. Tensorflow federated. 2019.

[4] The FATE Authors. Fate: An industrial grade federated learning framework. 2019.

[5] Sebastian Caldas, Sai Meher Karthik Duddu, Peter Wu, Tian Li, Jakub Konečný, H. Brendan McMahan, Virginia Smith, and Ameet Talwalkar. Leaf: A benchmark for federated settings. *arXiv preprint arXiv:1812.01097*, 2019.

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

[7] Yanjun Ma, Dianhai Yu, Tian Wu, and Haifeng Wang. Paddlepaddle: An open-source deep learning platform from industrial practice. *Frontiers of Data and Domputing*, 1(1):105, 2019.

[8] The clara training framework authors. Nvidia clara. 2019.

[9] Pitch Patarasuk and Xin Yuan. Bandwidth optimal all-reduce algorithms for clusters of workstations. *Journal of Parallel and Distributed Computing*, 69(2):117–124, 2009.

[10] John Lamping and Eric Veach. A fast, minimal memory, consistent hash algorithm. *arXiv preprint arXiv:1406.2294*, 2014.

[11] Pathum Chamikara Mahawaga Arachchige, Peter Bertok, Ibrahim Khalil, Dongxi Liu, Seyit Camtepe, and Mohammed Atiquzzaman. Local differential privacy for deep learning. *IEEE Internet of Things Journal*, 7(7):5827–5842, 2020.

[12] Oded Goldreich. Secure multi-party computation. *Manuscript. Preliminary version*, 1998.

[13] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. *Handbook of Systemic Autoimmune Diseases*, 1(4), 2009.

[14] The TensorFlow Federated Authors. Tensorflow federated stack overflow dataset. 2019.

[15] Chaoyang He, Songze Li, Jinhyun So, Xiao Zeng, Mi Zhang, Hongyi Wang, Xiaoyang Wang, Praneeth Vepakomma, Abhishek Singh, Hang Qiu, Xinghua Zhu, Jianzong Wang, Li Shen, Peilin Zhao, Yan Kang, Yang Liu, Ramesh Raskar, Qiang Yang, Murali Annavaram, and Salman Avestimehr. Fedml: A research library and benchmark for federated machine learning. *arXiv preprint arXiv:2007.13518*, 2020.

[16] Sashank Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečný, Sanjiv Kumar, and H. Brendan McMahan. Adaptive federated optimization. *arXiv preprint arXiv:2003.00295*, 2020.

[17] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. *arXiv preprint arXiv:1909.11942*, 2020.

[18] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2019.

[19] Zheng Wang, Xiaoliang Fan, Jianzhong Qi, Chenglu Wen, Cheng Wang, and Rongshan Yu. Federated learning with fair averaging. *arXiv preprint arXiv:2104.14937*, 2021.

[20] Jianyu Wang, Qinghua Liu, Hao Liang, Gauri Joshi, and H. Vincent Poor. Tackling the objective inconsistency problem in heterogeneous federated optimization. *Advances in Neural Information Processing Systems*, 33:7611–7623, 2020.

[21] 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, 2020.

[22] Yujun Lin, Song Han, Huizi Mao, Yu Wang, and William J. Dally. Deep gradient compression: Reducing the communication bandwidth for distributed training. *arXiv preprint arXiv:1712.01887*, 2020.

[23] Hanlin Tang, Shaoduo Gan, Ce Zhang, Tong Zhang, and Ji Liu. Communication compression for decentralized training. In *Advances in Neural Information Processing Systems*, 31, 2018.

[24] Constantin Philippenko and Aymeric Dieuleveut. Bidirectional compression in heterogeneous settings for distributed or federated learning with partial participation: tight convergence guarantees. *arXiv preprint arXiv:2006.14591*, 2021.

[25] Briland Hitaj, Giuseppe Ateniese, and Fernando Perez-Cruz. 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, 2017.

[26] Qi Zhao, Chuan Zhao, Shujie Cui, Shan Jing, and Zhenxiang Chen. Privatedl: Privacy reserving collaborative deep learning against leakage from gradient sharing. *International Journal of Intelligent Systems*, 35(8):1262–1279, 2020.

[27] Milad Nasr, Reza Shokri, and Amir Houmansadr. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In *2019 IEEE Symposium on Security and Privacy (SP)*, 739–753, 2019.

[28] A. Ghosh, J. Chung, Y. Dong, and K. Ramchandran. An efficient framework for clustered federated learning. In *Advances in Neural Information Processing Systems*, 33:19586-19597, 2020.

[29] Felix Sattler, Klaus-Robert Müller, and Wojciech Samek. Clustered Federated Learning: Model-Agnostic Distributed Multitask Optimization Under Privacy Constraints. *IEEE Transactions on Neural Networks and Learning Systems*, 32(8):3710–3722, 2021.

[30] Ming Xie, Guodong Long, Tao Shen, Tianyi Zhou, Xianzhi Wang, Jing Jiang, and Chengqi Zhang. Multi-Center Federated Learning. *arXiv preprint arXiv:2005.01026*, 2020.

[31] Yishay Mansour, Mehryar Mohri, Jae Ro, and Ananda Theertha Suresh. Three approaches for personalization with applications to federated learning. *arXiv preprint arXiv:2002.10619*, 2020.

[32] Tian Li, Shengyuan Hu, Ahmad Beirami, and Virginia Smith. Ditto: Fair and Robust Federated Learning Through Personalization. *International Conference on Machine Learning*, 6357–6368, 2021.

[33] Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. Personalized federated learning: A meta-learning approach. *arXiv preprint arXiv:2002.07948*, 2020.

[34] Yue Tan, Guodong Long, Lu Liu, Tianyi Zhou, Qinghua Lu, Jing Jiang, and Chengqi Zhang. Fedproto: Federated prototype learning over heterogeneous devices. *arXiv preprint arXiv:2105.00243*, 2021.

[35] Tao Lin, Lingjing Kong, Sebastian U. Stich, and Martin Jaggi. Ensemble Distillation for Robust Model Fusion in Federated Learning. *Advances in Neural Information Processing Systems*, 33:2351–2363, 2021.

[36] Mengwei Xu, Yuxin Zhao, Kaigui Bian, Gang Huang, Qiaozhu Mei, and Xuanzhe Liu. Federated Neural Architecture Search. *arXiv preprint arXiv:2002.06352*, 2020.