Neursafe-FL: A Reliable, Efficient, Easy-to-Use Federated Learning Framework

TANG Bo[1], ZHANG Chengming[1], WANG Kewen[1], GAO Zhengguang[2], HAN Bingtao[2]
(1. ZTE Corporation, Shenzhen 518057, China;
2. The State Key Laboratory of Mobile Network and Mobile Multimedia Technology, Shenzhen 518055, China)

DOI: 10.12142/ZTECOM.202203006
https://kns.cnki.net/kcms/detail/34.1294.TN.20220826.1322.001.html, published online August 26, 2022
Manuscript received: 2022-06-18

Abstract: Federated learning (FL) has developed rapidly in recent years as a privacy-preserving machine learning method, and it has been gradually applied to key areas involving privacy and security such as finance, medical care, and government affairs. However, the current solutions to FL rarely consider the problem of migration from centralized learning to federated learning, resulting in a high practical threshold for federated learning and low usability. Therefore, we introduce a reliable, efficient, and easy-to-use federated learning framework named Neursafe-FL. Based on the unified application program interface (API), the framework is not only compatible with mainstream machine learning frameworks, such as Tensorflow and Pytorch, but also supports further extensions, which can preserve the programming style of the original framework to lower the threshold of FL. At the same time, the design of componentization, modularization, and standardized interface makes the framework highly extensible, which meets the needs of customized requirements and FL evolution in the future. Neursafe-FL is already on Github as an open-source project[1].

Keywords: federated learning; privacy-preserving; Neursafe-FL

Citation (IEEE Format): B. Tang, C. M. Zhang, K. W. Wang, et al., "Neursafe-FL: a reliable, efficient, easy-to-use federated learning framework," *ZTE Communications*, vol. 20, no. 3, pp. 43 - 53, Sept. 2022. doi: 10.12142/ZTECOM.202203006.

1 Introduction

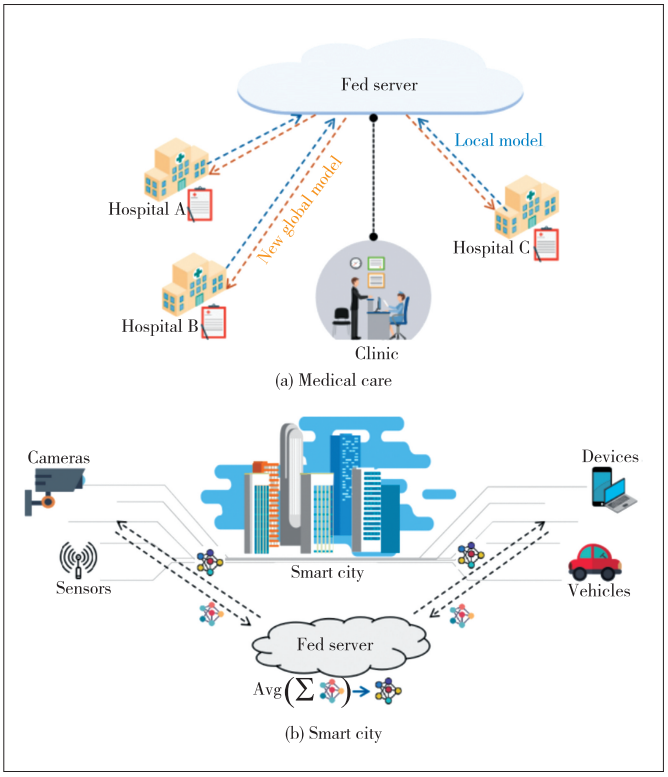
Federated learning (FL) is a distributed machine learning method that uses decentralized data residing on the client side to complete model training with the coordination of a central server[1-5]. It is a general method of "bringing the code to the data, instead of the data to the code"[3], and focuses on the security and privacy of data. Since the data are limited in the client domain during the training process and the intermediate data are encapsulated by the privacy-preserving algorithm, it could avoid privacy leakage in the training and inference process of machine learning. Especially with data protection laws and regulations, like the European General Data Protection Regulation (GDPR)[6], federated learning has been regarded as a hotspot in AI research.

Federated learning has many practical cases in the fields of finance[7], medical care[8], and smart city[9]. Fig. 1(a) shows the scene in medical care. It is expected to integrate the data of multiple hospitals to train a model, but the patient's information is very sensitive and private, which cannot be shared among hospitals. Fig. 1(b) shows that more and more intelligent edge devices with computing power, such as mobile phones, sensors, and cameras, join the smart city ecosystem. A large number of valuable data are generated on the devices. Since the data are private and impractical to collect, federated learning is very suitable for solving the problems in the above mentioned scenarios.

Federated learning breaks the data silos caused by privacy and security issues, which broadens the prospects of AI applications in many fields. However, FL introduces some unique challenges due to its distributed characteristics[10] shown as follows. 1) The problem of privacy leakage is caused by attack technologies such as membership inferring and feature inferring through intermediate data including model weights and gradients[11-13]; 2) The low efficiency of model convergence is caused by non-independent identically distribution (IID) data[14-15]; 3) The robustness of a model can be decreased by data poisoning[16], model poisoning[17] and other attack methods. There also exist efficiency problems which are caused by insufficient client-side computing power, data transmission bandwidth, etc[18].

We have developed a reliable, efficient, and easy-to-use open source framework to address the challenges mentioned

1. Neursafe-FL can be seen on the website of Github: https://github.com/neursafe/federated-learning.



▲ **Figure 1. Federated learning scenarios**

above in federated learning. Compared with the existing open source implementation for federated learning, more consideration is given to the migration of existing machine learning models from centralization to federation. The proposed framework cooperates well with mainstream machine learning frameworks and supports further upgradation, and it also retains the programming features of the original framework to simplify the implementations for various federated learning algorithms. Finally, the framework has advantages in future upgrades and evolution due to the componentization, modularization, and standardized interface.

The rest of this paper is organized as follows. Section 2 introduces the current research work on federated learning, including the technical challenges faced by federated learning, the comparison, and shortcomings of mainstream federated learning frameworks. In Section 3, we propose efficient and easy-to-use design solutions and principles. The design architecture, working principles, and the process of the system are introduced in Section 4. Section 5 presents experimental verification of multiple scenarios based on Neursafe FL. Finally, we summarize the contributions of this paper and point out directions for future work.

2 Related Work

With the increasing popularity of federated learning, a large amount of research work has been published to overcome the technical challenges of federated learning shown as follows. Differential privacy[19 – 23], secure multi-party computation[24 – 28], homomorphic encryption[29 – 33] and other privacy computing techniques were proposed to protect intermediate data such as the model weights and gradients, which avoids possible privacy leaks. Optimization and aggregation algorithms of FL such as FedAvg[1], FedNova[34], FedProx[35], SCAFFOLD[36], FedNas[37] were introduced to solve the problems of convergence efficiency caused by non-IID data. In Ref. [38 – 43], the authors proposed robust federated algorithms, such as Krum, FLRA, and Sageflow, to address the challenge of model robustness in the face of model attacks. The techniques of client-side incentives, quantization, models, and data compression were proposed to address the communication and computational efficiency problems in federated learning[44 – 53]. The incentive mechanism was adopted to solve the fairness problem in federated learning[54 – 56].

With the development of theoretical research on FL, a number of open source frameworks or libraries have emerged including Tensorflow Federated (TFF)[57], FATE[58], PySyft[59], FedML[60], and PaddleFL[61]. Among them, TFF, PySyft, and FedML are presented as the libraries for the research, while FATE and PaddleFL are frameworks or platforms for the production applications. However, these open source implementations have their own limitations as follows.

1) Most of the above work is developed based on a single underlying machine learning framework. For example, TFF is developed based on Tensorflow, FedML is implemented based on Pytorch, and PaddleFL is based on PaddlePaddle. Poorly substantial framework support leads to unnecessary costs for migrating FL applications.

2) The existing work supports limited application scenarios. For example, TFF only supports single-machine distributed simulation for research purposes; FATE and PaddleFL mainly solve cross-silo scenarios across data silos, while they are not suitable for cross-device scenarios. Although FedML supports a variety of application scenarios, the deployment process is very complicated. For example, the premise of FedML for various scenarios is that users must perform topology management, which makes user implementation more complex because network changes require the implementation change.

3) Most APIs of current frameworks are too complicated. Developers must learn proprietary API interfaces and programming specifications to implement federated learning, resulting in high costs for the migration of existing AI models.

4) Trade-offs between flexibility and usability are unsophisticated. On the one hand, some FL frameworks have a relatively high level of API encapsulation, which loses a certain degree of flexibility. On the other hand, the library represented by FedML has high flexibility, but the design makes development complicated, which leads to a high threshold for users.

Table 1 presents a comparison of open source projects. To solve the main challenges of federated learning mentioned

▼Table 1. Comparison of open source frameworks

Concerns	Features	TFF	PySyft	FedML	FATE	PaddleFL	Neursafe-FL
Supported running mode	Standalone	√	√	√	√	√	√
	Cross-device	×	×	√	×	×	√
	Cross-silo	×	×	√	√	√	√
Aggregation algorithms	IID (FedAvg, etc.)	√	√	√	√	√	√
	Non-IID (FedProx, etc.)	×	×	√	√	-	√
Supported underlying framework	Tensorflow	√	√	×	√	×	√
	Pytorch	×	√	√	√	×	√
Privacy protection methods	DP	√	√	√	×	√	√
	MPC	×	√	×	√	√	√
	HE	×	√	×	√	×	×
Flexibility	Device management	×	×	×	×	×	√
	Customization	×	×	√	×	×	√

DP: differential privacy FL: federated learning HE: homomorphic encryption IID: independent identically distribution MPC: multi-party computation TFF: tensorflow federated

above, Neursafe-FL is proposed as an efficient, simple, and easy-to-use federated learning framework without losing flexibility.

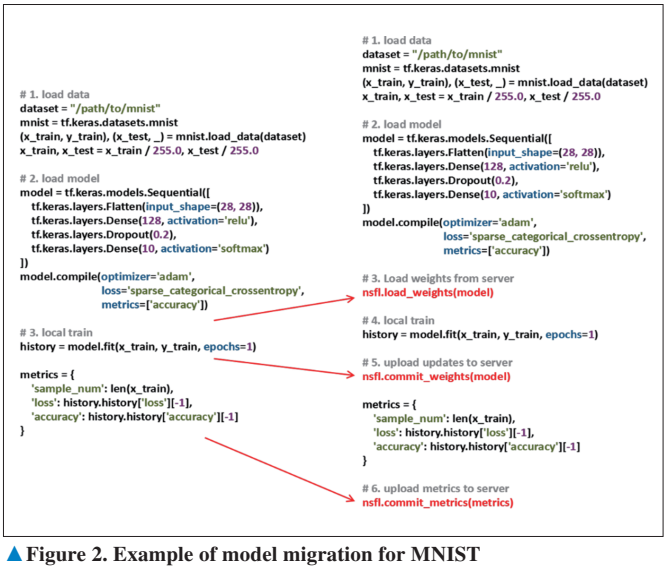
3 Design of Neursafe-FL

Neursafe-FL adopts the principles of componentization and modularization in the design. According to different functions and characteristics, we divide the system into several components and modules such as job scheduling, client management and selection, privacy protection, and optimization aggregation. The components are decoupled to reduce system complexity and provide feature-level scalability. Through componentized design, Neursafe-FL enables reliable services based on microservice management, the high availability (HA) mechanism, and job-level fault tolerance processing. In order to improve the usability and meet the requirements of long-term evolution for federated learning, we have made more efforts in the following areas:

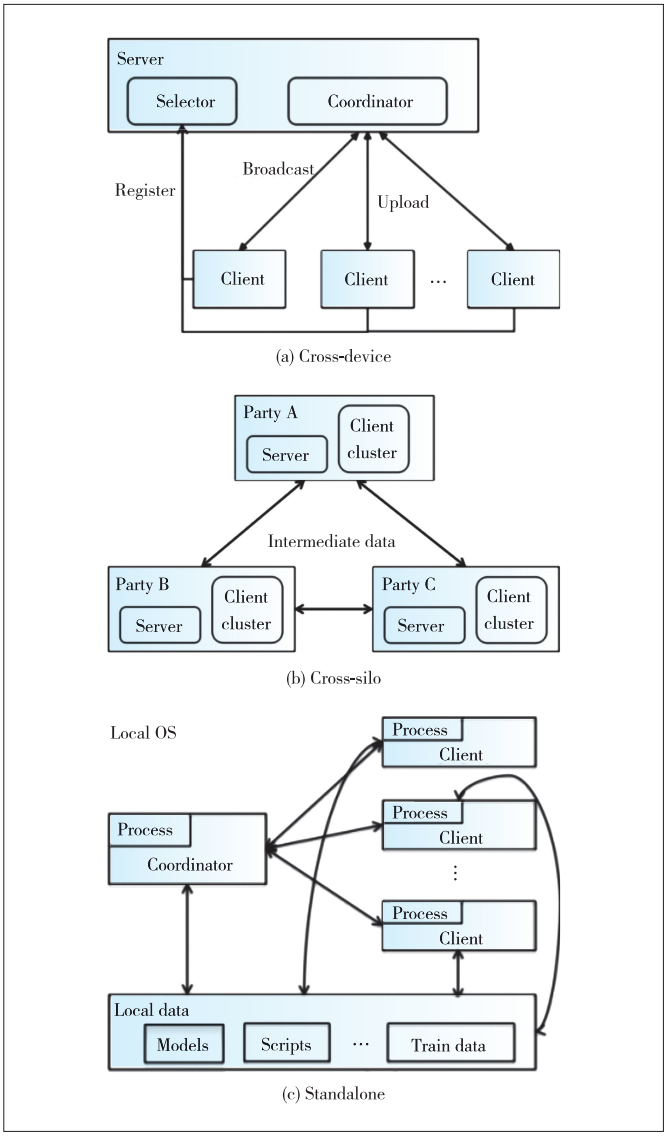
•Portability: There are a large number of existing center-based learning programs to be migrated to federated learning programs. Therefore, it is valuable to simplify the FL migration and even complete the migration with zero coding. To achieve this goal, Neursafe-FL has the following designs: 1) A minimalist unified API design is adopted; 2) On the basis of the unified API, it supports mainstream machine learning frameworks that support Tensorflow and Pytorch currently, and it also supports new frameworks by implementing the corresponding interfaces. In this way, it retains the programming style of the original framework, which significantly simplifies the program development of FL. Fig. 2 is an example of the FL migration of training on MNIST.

•Multi-running mode: Neursafe-FL supports standalone modes for research purposes. In this scenario, Neursafe-FL only deploys the server-side coordinator and one or more client processes on a single node for distributed simulation. For cross-silo and cross-device, we comprehensively consider the compatibility of two running modes in client management and scheduling design. In the cross-device mode, it enables clients to join in and log out of the system by registering and quitting, and provides a set of client selection algorithms with extension capabilities, which meets the requirements of training efficiency and model robustness[3]. There are fewer participants in the cross-silo mode, where clients can join the system by configuration, and can be selected by configuration or label matching. At the same time, each participant can deploy the server and client simultaneously, and the client also supports multi-task parallelism. Figs. 3(a), 3(b), and 3(c) are examples of the above running mode respectively.

•Extensibility: Federated learning is a fast-growing field, with new requirements and more advanced algorithms emerging. It requires the system with a high degree of flexibility, which may rise complexity in use. To trade off the flexibility and usability, we provide a user-friendly API for normal users, and an advanced interface for researchers to make further ex-



▲Figure 2. Example of model migration for MNIST



▲ **Figure 3. Multi-running mode**

tensions. In the second case, Neursafe-FL provides two approaches for the extension. 1) It extends the standardized algorithm interface and integrates it into the system as a part of the framework. 2) It extends through the callback interface for more customized requirements, but only one callback is validated at the same time. A detailed description of the extensibility is shown as follows.

• Extension of security algorithms: Users can implement new security and privacy algorithms through the form of libraries. To facilitate this form of extension, we provide the standardized interfaces such as "encrypt" and "decrypt", and decouple the security algorithms from federated learning processes. Users can integrate the security algorithm into the federated training process by extending the above standardized interface and referencing it in the configuration file. Based on the standard interface, the framework provides security algorithms such as differential privacy and secret sharing aggregation, and users can extend other security algorithms, homomorphic encryption, and secure multi-party computation as well.

• Extension of client selection algorithms: In a scenario with a large number of devices, selecting unreasonable training devices leads to resource mismatch, unfairness, and stragglers. Therefore, two expansion interfaces are provided for pre-selection and on-selection during the client selection process. Users can add filtering rules through the pre-selection interface, and prioritize qualified devices through the on-selection interface. For example, users can add trusted device matching rules through the pre-selected interface to filter out untrusted devices in the case of malicious parties. They can also use priority scores through an on-selection interface to select a more stable and reliable device based on the computing resources of the device, network status, and other information. We have provided rules for tag matching, resource matching, performance priority, and data priority. Users can add custom rules according to the above extension interface in two ways: by file and by webhook.

• Extension of aggregation algorithms: Aggregation algorithms have different effects in different scenarios. For example, the traditional FedAvg algorithm cannot face the challenges such as data heterogeneity and imbalance. Therefore, we provide two ways to extend the aggregation algorithm. 1) We inherit the base class of aggregator and integrate it directly into the framework; 2) Through the callback interface, we abstract the training process of federated learning into three steps: server-side broadcast, client-side reporting, and server-side aggregation, corresponding to the callback interfaces to broadcast custom data, aggregate updates on the server side, and process the final result. Users can implement the callback functions and validate them in the coordinator's configuration file. For example, we implement the SCAFFOLD aggregation algorithm with the second method to solve the problem in the non-IID scenario.

• Extension of machine learning framework: In order to be compatible with different machine learning frameworks such as Tensorflow, Pytorch, and Caffe, we encapsulate the framework and provide a unified standard interface for the upper layer. To support a new machine learning framework, users just need to complete the following adaptations through the standard interface: 1) adaptation of model operations such as loading, saving, and evaluation; 2) adaptation of weight operations, such as weight acquisition, weight assignment, and weight operations; 3) preprocessing operations on datasets for model evaluation and verification; 4) implementation of some security and privacy interfaces. An adapted ML framework can be integrated into the federated learning framework and run by the configuration parameters. The framework currently supports Tensorflow and Pytorch, and users can also extend the support of other machine learning frameworks according to the above steps.

4 Architecture of Neursafe-FL

The architecture of Neursafe-FL is shown in Fig. 4, and the cooperation between the components is shown in Fig. 5. Core components of Neursafe-FL are presented as follows.

•Infrastructure layer: Neursafe-FL has completed the adaptation at the infrastructure layer. On the server side, we deploy it on the Container as a Service (CaaS), which is Kubernetes by default. High reliability of FL core component services is guaranteed through the HA mechanism of CaaS. In the cross-device scenario, the client side supports two process modes: native OS process or containerized process. Containerized deployment improves system portability. In the cross-silo scenario, CaaS deployment is still used on the client side, which enables better parallel scheduling of tasks.

•Job scheduler: It is the core component of job management and scheduling. We schedule jobs according to job queuing and the current system resource status. The scheduling algorithm needs to consider the efficiency and fairness of the system resource when satisfying job requirements.

•Coordinator: It is dynamically created by the job scheduler for each job. It is responsible for the organization and coordination of federated training, including client selection, task dispatching, and model aggregation.

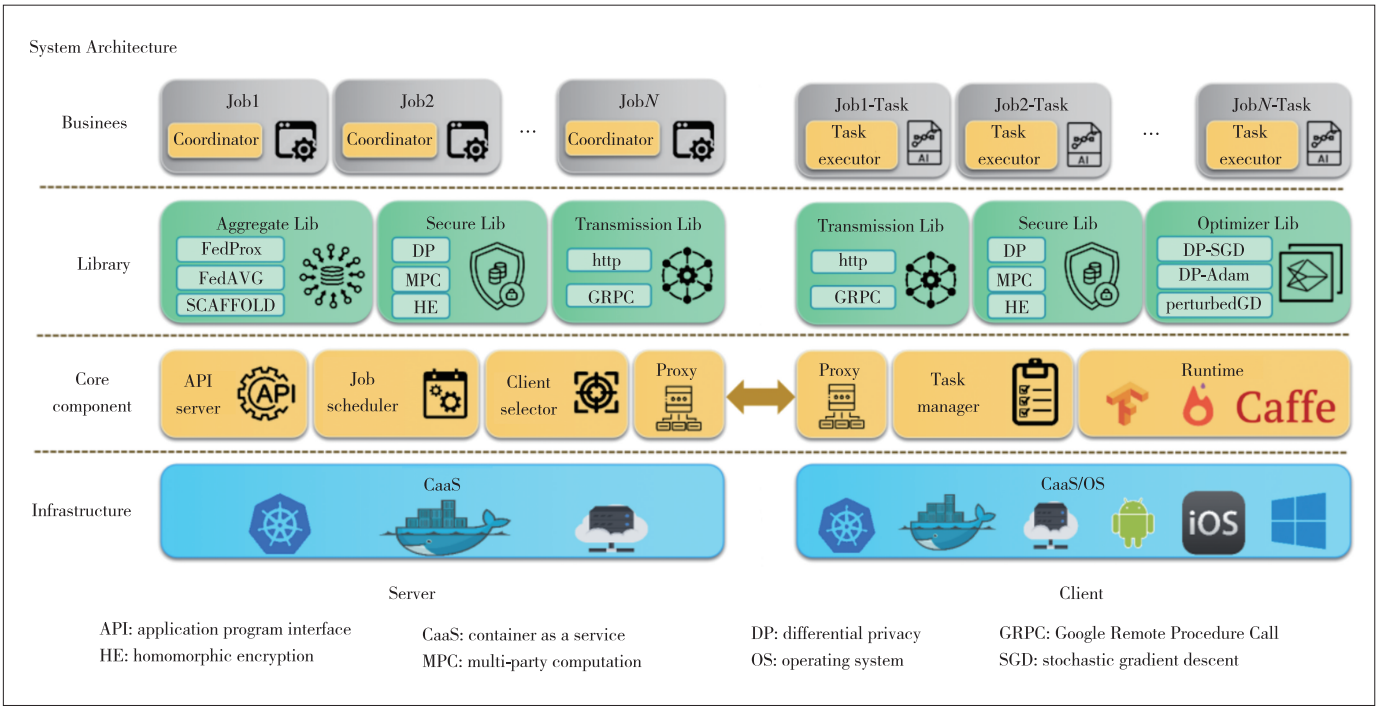
•Client selector: It is responsible for managing clients and responding to client selection requests. A client selector supports clients to join the system by registration or configuration. The client selection algorithm includes filtering and prioritization. The basic filters include whether there is a required dataset, whether it supports the specified machine learning framework (Runtime) or operating systems, etc., and whether it supports the extension of the filtering algorithm. The basic priorities include the number of data, computing power, parallelism, bandwidth, etc. The scalability of the client selection algorithm is expected to meet the needs of federated learning for client selection in terms of efficiency, robustness, and fairness[42 – 45].

• Task manager: It is the daemon component of the client. The main functions include the client's resources and status reporting, responding to the task issued by the server, and completing the task scheduling. In the cross-silo scenario, the client needs to execute tasks concurrently, and the task manager needs to schedule tasks according to the task queue and its local resource status.

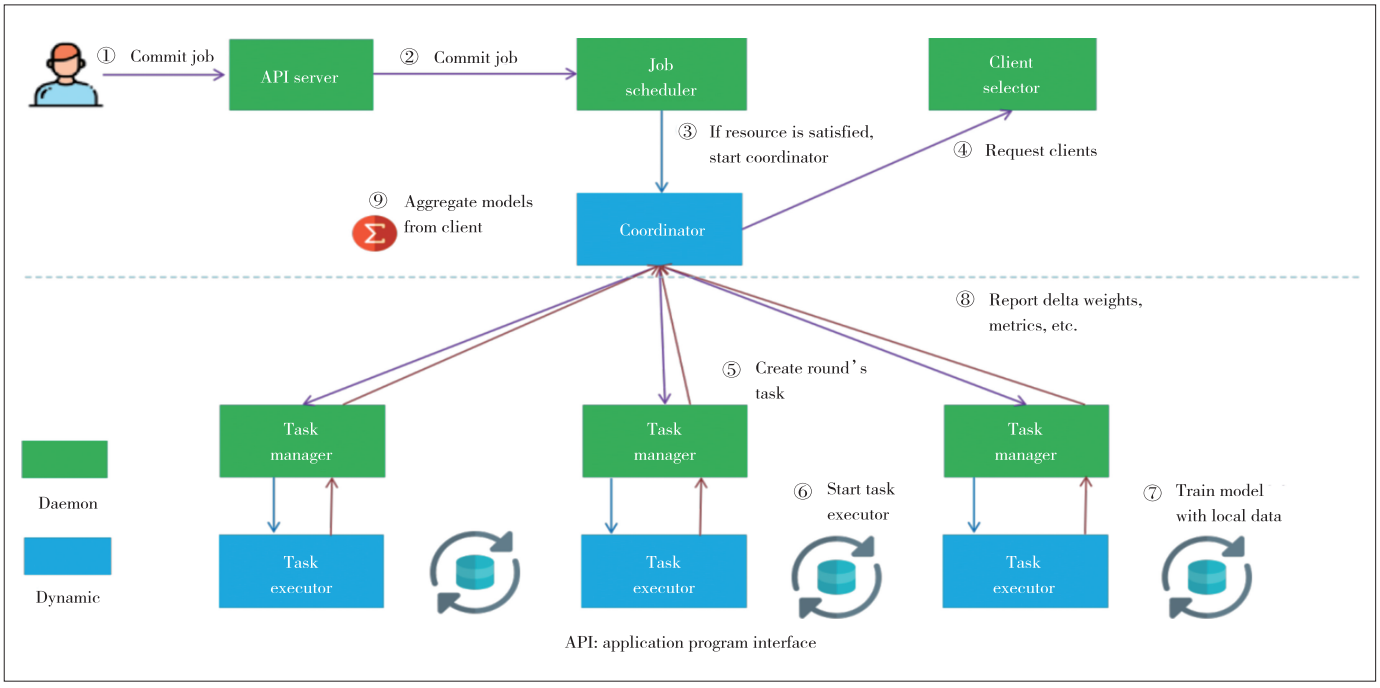
• Task executor: The client-side local executor of the federated task, which is dynamically created by the task manager when it receives tasks issued by the server. One task executor only manages one task to ensure the decoupling between tasks.

• Algorithms and basic libraries: Neursafe-FL encapsulates privacy security algorithms, federated optimization and aggregation algorithms, federated robustness algorithms, and low-level communications to simplify frameworks and application development. Users can extend the algorithm through the standardized algorithm interface, and benefit from the loose coupling between the algorithm and the framework.

Fig. 5 illustrates the interaction between Neursafe-FL core components in the job scheduling procedure as an example. 1) The user submits a job request to the job scheduler via the API server. 2) The job scheduler starts the coordinator when the system resources are satisfied. 3) The coordinator starts



▲**Figure 4. Architecture of Neursafe-FL**



▲**Figure 5. Interaction between Neursafe-FL core components**

the job execution process and requires clients to participate in federated training from the client selector. 4) The coordinator sends federated training tasks to clients. 5) After receiving the federated task, the client dynamically starts the local task executor. 6) The task executor uses local data to train the model. 7) Clients submit the model weight delta and the measurement generated by local training to the job coordinator. 8) The coordinator aggregates the client models to obtain a new global model. The coordinator decides to finish the federated training according to various criteria, such as model convergence and the max number of rounds reached.

5 Results of Neursafe-FL

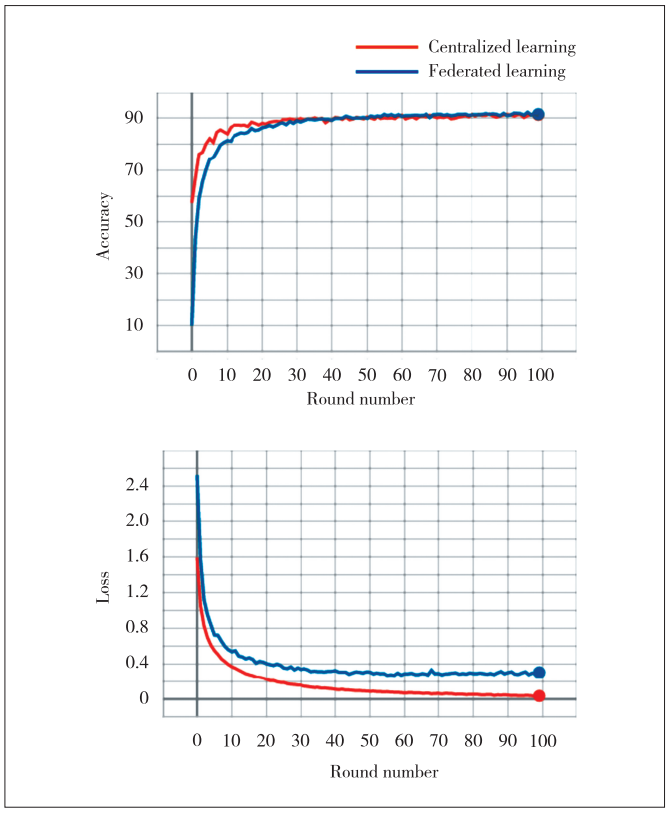
In order to evaluate the performance of Neursafe-FL, we design the following experiments: the comparison of convergence efficiency and accuracy of centralized learning and federated learning, the convergence comparison of federated training with different client numbers, the comparison of convergence efficiency of different federated aggregation algorithms under non-IID data, and the impact of privacy security algorithms on model accuracy and training efficiency. All experiments adopt a CNN model on two datasets: MNIST and CIFAR10.

The experiment in Fig. 6 is based on the CIFAR10 dataset. Three clients participate in federated training and the sample data for federated training are split according to the IID method. In order to compare the convergence efficiency, we set the client to perform only one epoch iteration per round. The experimental results in Fig. 6 show that the convergence efficiency and model accuracy of federated training and centralized training are almost the same under the IID data.

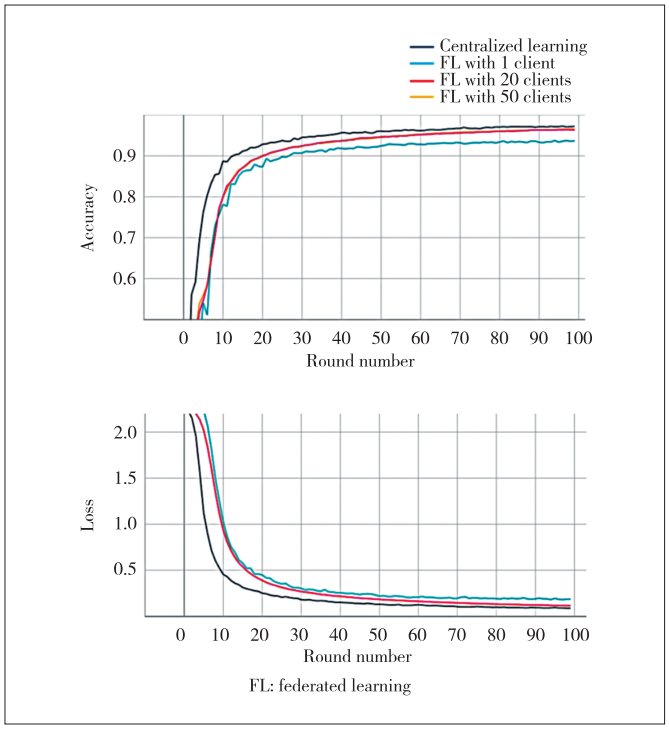
In Fig. 7, we compare the convergence performance of federated learning and centralized learning in terms of loss and accuracy. Data are evenly distributed to different clients by IID sampling. We test the scenario of federated learning with 1 client, 20 clients, and 50 clients. From the convergence curve of loss and accuracy in Fig. 7, the convergence of centralized learning is better than that of federated learning. On the other hand, the convergence performance of federated learning with multiple clients is significantly better than that of a single client, and it is close to the centralized learning. It can be seen that the convergence rate of federated learning and centralized learning are consistent. In addition, federated learning can ensure the privacy and security of clients.

Five federated clients participate in federated training, and the data are split in an extremely unbalanced manner, in which there are no samples with the same label among clients. Three federated aggregation algorithms, FedAvg, FedProx and SCAFFOLD, are used in the experiment. The results in Fig. 8 show that FedAvg is difficult to converge under this extreme data distribution, while FedProx and SCAFFOLD can converge with similar convergence efficiency. The results prove that it's necessary to select appropriate federated optimization and aggregation algorithms according to different data distributions to ensure the convergence efficiency of the model.

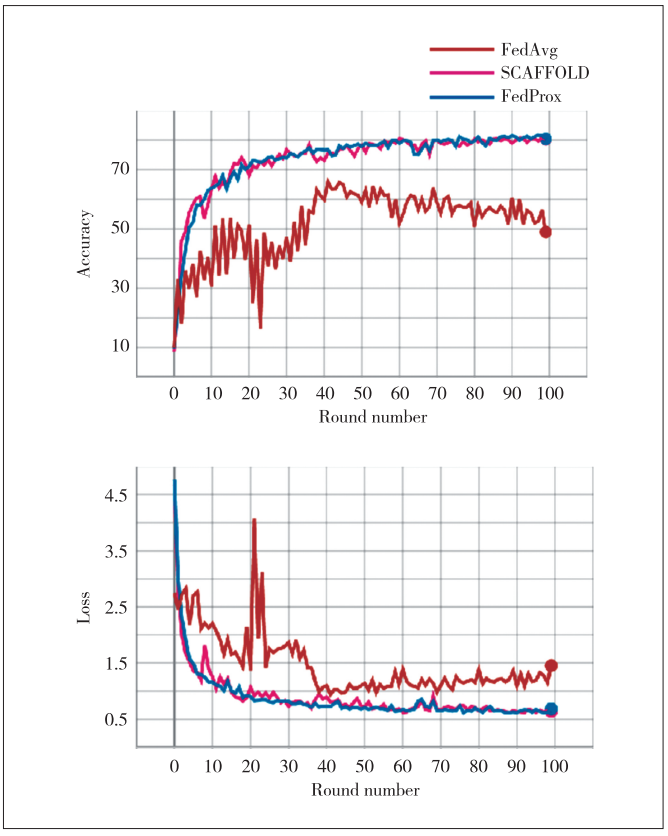
Two security algorithms, differential privacy and secret share aggregation (SSA) are tested in terms of overhead, model accuracy, and convergence efficiency. Both algorithm experiments are based on the MNIST and CIFAR10 datasets. The impact of differential privacy on the accuracy of a model is evaluated in the experiment. The results are shown in Fig. 9.



▲Figure 6. Comparison of federated training and centralized training



▲Figure 7. Convergence comparison of federated training with different client numbers



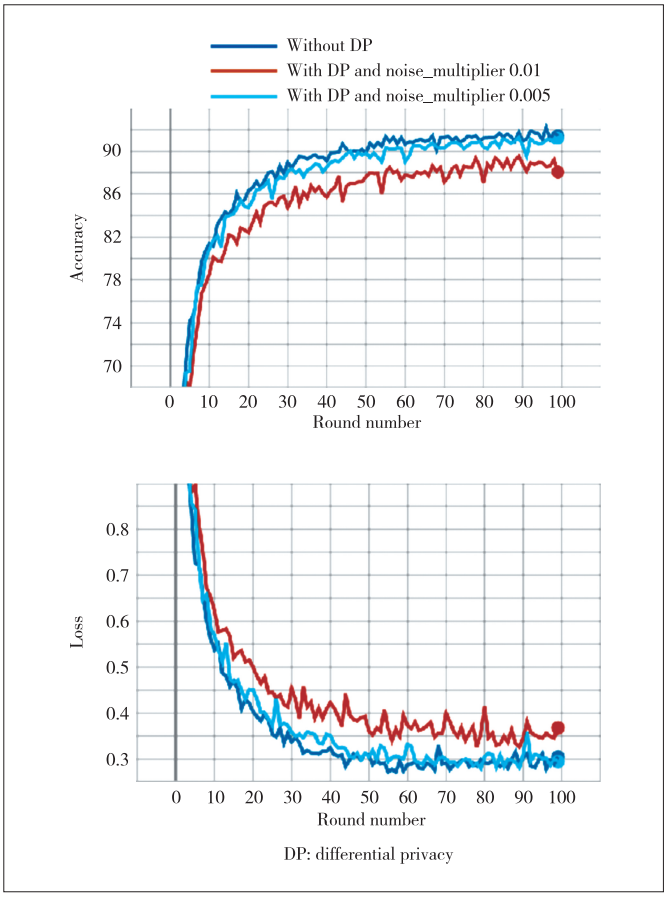
▲Figure 8. Results of different aggregation algorithms under non-IID data

After 100 rounds of training, the model accuracy with differential privacy is 0.912, while the accuracy of the model without differential privacy is 0.914. The results show that differential privacy has a limited impact on the model's accuracy. The overhead of differential privacy is shown in Table 2. Compared with the overall overhead of federated training, the overhead of differential privacy accounts for a small proportion. Neursafe-FL reduces the times of adding noise by adding corresponding noise to the updated weights of the client completing one round of model training. The budget of Neursafe-FL is much smaller than that of the method of adding noise during gradient update.

A comparative test of introducing SSA and not introducing SSA is conducted, and the results in Fig. 10 show that the curves of convergence and accuracy are completely consistent. The SSA algorithm based on the principle of cryptography is lossless. Therefore, the SSA-based algorithm has no effect on the convergence and accuracy of federated training.

The SSA algorithm is tested in the scenario where the client is disconnected. The results in Fig. 11 show that the convergence curve is almost the same as that in the normal scenario, and the SSA algorithm remains lossless. The SSA algorithm randomly selects disconnected clients every round, which only causes slight differences in the disconnected case.

The impact of the number of federated training clients on the SSA algorithm is studied in the experiment, and we analyze the



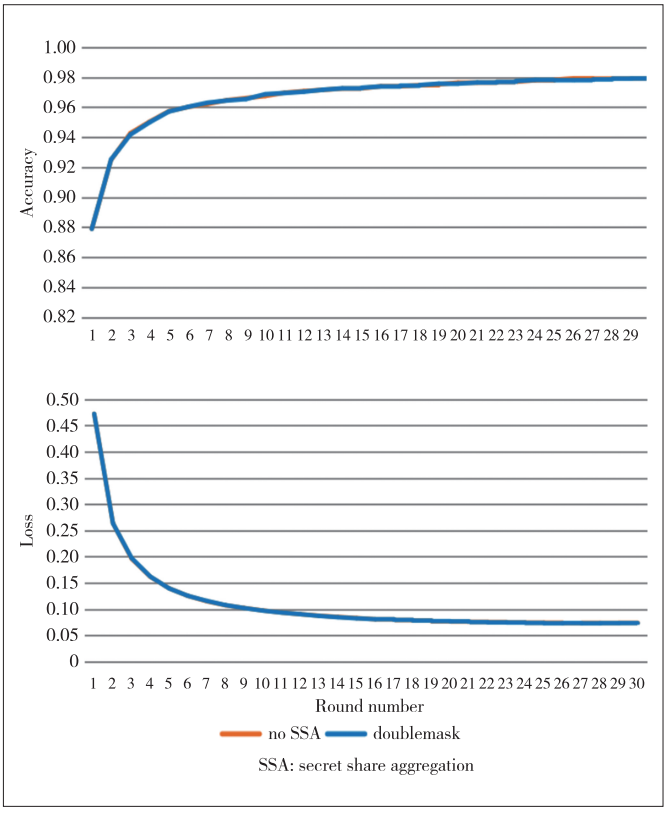
▲Figure 9. A comparative test of introducing DP and not introducing differential privacy

▼Table 2. Results of different parameters for differential privacy

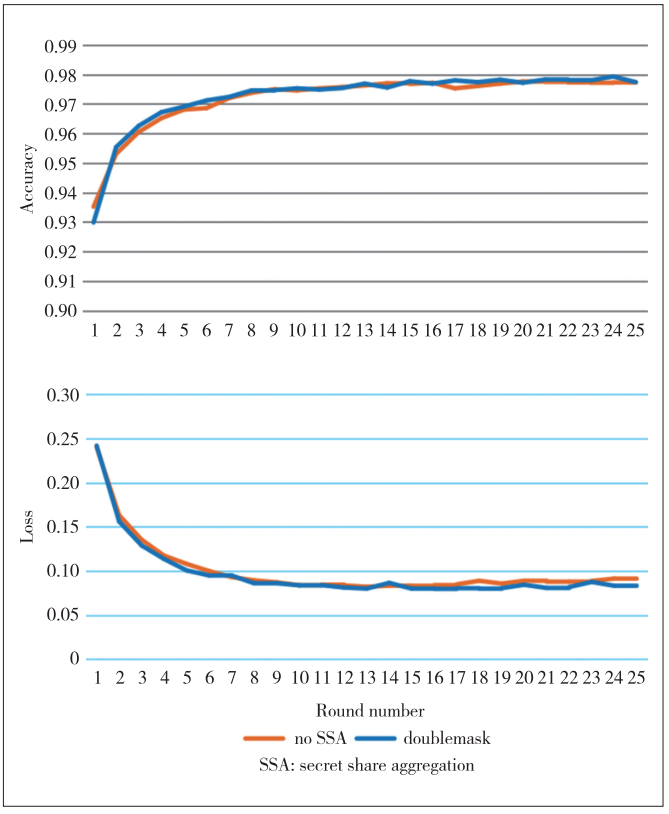
Type	Round Number	Noise_multiplier	Time Cost/s	Accuracy
Without DP	100	-	2 873.22	0.914 3
	50	-	1 396.56	0.898 4
With DP	100	0.01	2 879.36	0.912 2
	100	0.005	2 882.86	0.891 0
	50	0.01	1 390.42	0.876 3

DP: differential privacy

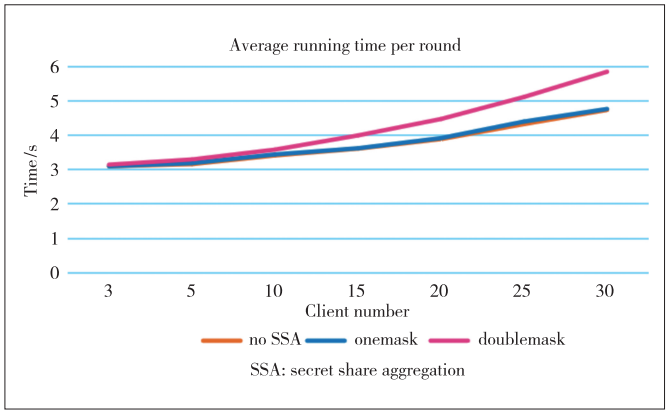
effect of clients' number on the overhead of federated training in the models without using the SSA algorithm, using the SSA onemask, and using the SSA doublemask respectively. The results are shown in Fig. 12. In the one-mask mode, the performance of SSA is basically the same as the performance when SSA is not used. The number of clients nearly has no impact on it, because each client only needs to send its own Diffie-Hellman (DH) public key to other clients under the one-mask model. This process is performed concurrently with model training, and the performance is not affected. In the double-mask mode, the performance of SSA increases with the number of clients. When the results are aggregated, additional communication is required to recover the mask. Therefore, the perfor-



▲Figure 10. A comparative test of federated training with SSA and without SSA



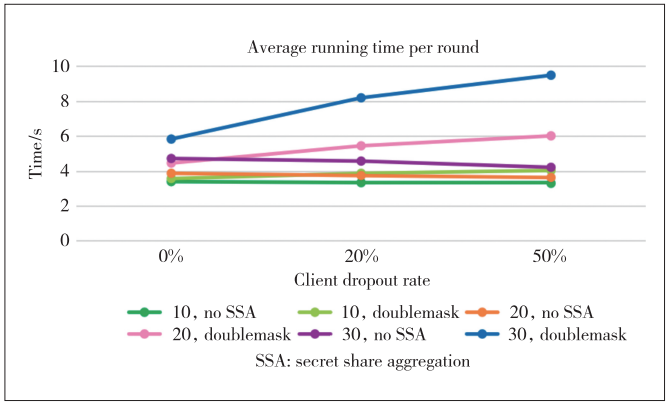
▲Figure 11. Impacts of SSA in the scenario where the clients are disconnected



▲**Figure 12. Impacts of client's number on the performance of SSA**

mance overhead will also increase with the increase in the number of clients.

The SSA algorithm is tested in the case of different drop rates of clients participating in the federated training, and we analyze the effect of different drop rates on the performance of federated training without using the SSA algorithm, using the SSA one-mask, and using the SSA double-mask respectively in Fig. 13. In the figure, "10, no SSA" denotes the training with 10 clients participating without using SSA algorithm, and the other abbreviations can be translated similarly. As shown in the figure, the overhead of using SSA increases substantially as the drop rate increases, and as the number of dropout clients increases, the overhead also increases. The overhead increases, because it requires the SSA algorithm to process the mask related to the disconnected clients, which leads to additional communication.



▲**Figure 13. Impacts of different dropout rates on the performance of federated training**

It can be seen that the two privacy security algorithms have their own advantages. Differential privacy introduces noise, which has an impact on the convergence of model training, but the impact is relatively small for the overhead. On the other hand, SSA can guarantee secure aggregation as a cryptographic algorithm without accessing the original data. This is a lossless algorithm, which ensures the accuracy and convergence of the model compared with the differential privacy, but it introduces a large overhead in calculation and transmission. Therefore, users can choose different security algorithms according to the requirements of the accuracy, calculation, and transmission for the applications.

6 Conclusions and Future Work

This paper introduces a new federated learning framework: Neursafe-FL, which focuses on the main challenges of federated learning, such as privacy, security, efficiency, and robustness, and on the usability to reduce the cost of model migration. In the design, we simplify the API, make it compatible with multiple mainstream machine learning frameworks, and enable framework extensions. All of these designs lower the threshold for federated learning. Through componentization, modulization, and standardization design, the scalability of the system is improved to meet the diverse needs of users. Experiments show that this framework has a good performance in terms of model convergence and accuracy under various settings.

In the future, we will focus on the following aspects: 1) Integrating more algorithms into Neursafe-FL to improve the efficiency, security, and robustness of federated learning. 2) Supporting vertical and transfer federation to meet the needs of different data distribution scenarios. 3) Enriching system features such as enabling more infrastructure and OS, and expanding support for more machine learning frameworks, federated inference, etc., to facilitate more FL applications.

References

[1] MCMAHAN B, MOORE E, RAMAGE D, et al. Communication-efficient learning of deep networks from decentralized data [C]//International Conference on Artificial Intelligence and Statistics. PMLR, 2017: 1273 – 1282
[2] YANG Q, LIU Y, CHEN T J, et al. Federated machine learning: concept and applications [EB/OL]. [2022-04-01]. https://arxiv.org/abs/1902.04885
[3] BONAWITZ K, EICHNER H, GRIESKAMP W, et al. Towards federated learning at scale: system design [EB/OL]. [2022-04-01]. https://arxiv. org/abs/1902.01046
[4] KAIROUZ E B P, MCMAHAN H B. Advances and open problems in federated learning [J]. Foundations and trends in machine learning, 2021, 14(1): 1 – 21. DOI: 10.1561/2200000083
[5] LI T, SAHU A K, TALWALKAR A, et al. Federated learning: challenges, methods, and future directions [J]. IEEE signal processing magazine, 2020, 37(3): 50 – 60. DOI: 10.1109/MSP.2020.2975749
[6] VOIGT P, VON DEM BUSSCHE A. The EU general data protection regulation (GDPR): a practical guide [M]. Cham: Springer International Publishing, 2017
[7] LONG G D, TAN Y, JIANG J, et al. Federated learning for open banking federated learning [J]. Federated Learning 2020: 240 – 254. DOI: 10.1007/978-3-030-63076-8_17
[8] XU J, GLICKSBERG B S, SU C, et al. Federated learning for healthcare informatics [J]. Journal of healthcare informatics research, 2021, 5(1): 1 – 19. DOI: 10.1007/s41666-020-00082-4

[9] JIANG J C, KANTARCI B, OKTUG S, et al. Federated learning in smart city sensing: challenges and opportunities [J]. Sensors, 2020, 20(21): 6230. DOI: 10.3390/s20216230
[10] LYU L, YU H, YANG Q. Threats to federated learning: a survey [EB/OL]. [2022-04-01]. https://deepai.org/publication/threats-to-federated-learning-a-survey
[11] BAGDASARYAN E, VEIT A, HUA Y, et al. How to backdoor federated learning [C]//International Conference on Artificial Intelligence and Statistics. PMLR, 2020: 2938 – 2948
[12] WANG Z B, SONG M K, ZHANG Z F, et al. Beyond inferring class representatives: user-level privacy leakage from federated learning [C]//IEEE Conference on Computer Communications. IEEE, 2019: 2512 – 2520. DOI: 10.1109/INFOCOM.2019.8737416
[13] NASR M, SHOKRI R, HOUMANSADR A. Comprehensive privacy analysis of deep learning: passive and active white-box inference attacks against centralized and federated learning [C]//IEEE Symposium on Security and Privacy. IEEE, 2019: 739 – 753. DOI: 10.1109/SP.2019.00065
[14] ZHAO Y, LI M, LAI L, et al. Federated learning with non-IID [EB/OL]. (2018-06-02) [2022-04-01]. https://arxiv.org/abs/1806.00582
[15] LI X, HUANG K, YANG W, et al. On the convergence of FedAvg on non-IID data [EB/OL]. (2020-06-25) [2022-04-01]. https://arxiv.org/abs/1907.02189
[16] ZHANG J L, CHEN J J, WU D, et al. Poisoning attack in federated learning using generative adversarial nets [C]//18th IEEE International Conference on Trust, Security and Privacy in Computing and Communications. IEEE, 2019: 374 – 380. DOI: 10.1109/TrustCom/BigDataSE.2019.00057
[17] FANG M, CAO X, JIA J, et al. Local model poisoning attacks to byzantine-robust federated learning [C]//29th USENIX Security Symposium. USENIX, 2020: 1605 – 1622
[18] REISIZADEH A, TZIOTIS I, HASSANI H, et al. Straggler-resilient federated learning: leveraging the interplay between statistical accuracy and system heterogeneity [EB/OL]. [2022-04-01]. https://arxiv.org/abs/2012.14453
[19] DWORK C. Differential privacy: a survey of results [C]//International Conference on Theory and Applications of Models of Computation. TAMC, 2008: 1 – 19
[20] MCMAHAN H B, ANDREW G, ERLINGSSON U, et al. A general approach to adding differential privacy to iterative training procedures [EB/OL]. [2022-04-01]. https://arxiv.org/abs/1812.06210
[21] ABADI M, CHU A, GOODFELLOW I, et al. Deep learning with differential privacy [C]//ACM SIGSAC Conference on Computer and Communications Security. ACM, 2016: 308 – 318
[22] DWORK C, ROTH A. The algorithmic foundations of differential privacy [J]. Foundations and trends in theoretical computer science, 2014, 9(3 – 4): 211 – 407
[23] BU Z, DONG J, LONG Q, et al. Deep learning with gaussian differential privacy [J]. Harvard data science review, 2020, 23. DOI: 10.1162/99608f92.cfc5dd25
[24] GOLDREICH O. Secure multi-party computation [EB/OL]. [2022-04-01]. https://www.wisdom.weizmann.ac.il/~oded/pp.html
[25] DU W, ATALLAH M J. Secure multi-party computation problems and their applications: a review and open problems [C]//Proceedings of the 2001 Workshop on New Security Paradigms. ACM, 2001: 13 – 22
[26] MOHASSEL P, RINDAL P. ABY3: a mixed protocol framework for machine learning [C]//Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. ACM: 2018: 35 – 52. DOI: 10.1145/3243734.3243760
[27] CHEN V, PASTRO V, RAYKOVA M. Secure computation for machine learning with SPDZ [EB/OL]. [2022-04-01]. https://arxiv.org/abs/1901.00329
[28] BONAWITZ K, IVANOV V, KREUTER B, et al. Practical secure aggregation for privacy-preserving machine learning [C]//Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. ACM, 2017. DOI: 10.1145/3133956.3133982
[29] GENTRY C. Fully homomorphic encryption using ideal lattices [C]//Proceedings of the 41st Annual ACM Symposium on Theory of Computing. ACM, 2009: 169 – 178. DOI: 10.1145/1536414.1536440
[30] GENTRY C. A fully homomorphic encryption scheme [M]. Stanford, USA: Stanford University, 2009
[31] ACAR A, AKSU H, ULUAGAC A S, et al. A survey on homomorphic encryption schemes [J]. ACM computing surveys, 2019, 51(4): 1 – 35. DOI: 10.1145/3214303
[32] ZHANG C, LI S, XIA J, et al. BatchCrypt: efficient homomorphic encryption for cross-silo federated learning [C]//2020 USENIX Annual Technical Conference. USENIX, 2020: 493 – 506
[33] FANG H K, QIAN Q. Privacy preserving machine learning with homomorphic encryption and federated learning [J]. Future Internet, 2021, 13(4): 94. DOI: 10.3390/fi13040094
[34] WANG J, LIU Q, LIANG H, et al. Tackling the objective inconsistency problem in heterogeneous federated optimization [J]. Advances in neural information processing systems, 2020, 33: 7611 – 7623
[35] LI T, SAHU A K, ZAHEER M, et al. Federated optimization in heterogeneous networks [EB/OL]. [2022-04-01]. https://arxiv.org/abs/1812.06127
[36] KARIMIREDDY S P, KALE S, MOHRI M, et al. SCAFFOLD: stochastic controlled averaging for on-device federated learning [EB/OL]. [2022-04-01]. https://arxiv.org/abs/1910.06378
[37] HE C Y, ANNAVARAM M, AVESTIMEHR S. Towards non-IID and invisible data with FedNAS: federated deep learning via neural architecture search [EB/OL]. [2022-04-01]. https://arxiv.org/abs/2004.08546
[38] BLANCHARD P, EL MHAMDI E M, GUERRAOUI R, et al. Machine learning with adversaries: Byzantine tolerant gradient descent [C]//The 31st International Conference on Neural Information Processing Systems. ACM, 2017
[39] REISIZADEH A, FARNIA F, PEDARSANI R, et al. Robust federated learning: the case of affine distribution shifts [J]. Advances in neural information processing systems, 2020, 33: 21554 – 21565
[40] PARK J, HAN D J, CHOI M, et al. Sageflow: robust federated learning against both stragglers and adversaries [J]. Advances in neural information processing systems, 2021, 34: 840 – 851
[41] PILLUTLA K, KAKADE S M, HARCHAOUI Z. Robust aggregation for federated learning [J]. IEEE transactions on signal processing, 2022, 70: 1142 – 1154. DOI: 10.1109/TSP.2022.3153135
[42] LI S, CHENG Y, WANG W, et al. Learning to detect malicious clients for robust federated learning [EB/OL]. [2022-04-01]. https://arxiv.org/abs/2002.00211
[43] SATTLER F, WIEDEMANN S, MULLER K R, et al. Robust and communication-efficient federated learning from non-IID data [J]. IEEE transactions on neural networks and learning systems, 2020, 31(9): 3400 – 3413. DOI: 10.1109/TNNLS.2019.2944481
[44] NISHIO T, YONETANI R. Client selection for federated learning with heterogeneous resources in mobile edge [C]//2019 IEEE International Conference on Communications. IEEE, 2019 : 1 – 7. DOI: 10.1109/ICC.2019.8761315
[45] GEYER R C, KLEIN T, NABI M. Differentially private federated learning: A client level perspective [EB/OL]. [2022-04-01]. https://arxiv. org/abs/1712.07557
[46] KANG J W, XIONG Z H, NIYATO D, et al. Incentive design for efficient federated learning in mobile networks: a contract theory approach [C]//2019 IEEE VTS Asia Pacific Wireless Communications Symposium. IEEE, 2019: 1 – 5. DOI: 10.1109/VTS-APWCS.2019.8851649
[47] CHEN W, HORVATH S, RICHTARIK P. Optimal client sampling for federated learning [EB/OL]. [2022-04-01]. https://arxiv.org/abs/2010.13723
[48] KONEČNÝ J, MCMAHAN H B, YU F X, et al. Federated learning: strategies for improving communication efficiency [EB/OL]. [2022-04-01]. https://arxiv.org/abs/1610.05492
[49] TRAN N H, BAO W, ZOMAYA A, et al. Federated learning over wireless networks: optimization model design and analysis [C]//IEEE Conference on Computer Communications. IEEE, 2019: 1387 – 1395. DOI: 10.1109/INFOCOM.2019.8737464
[50] GUHA N, TALWALKAR A, SMITH V. One-shot federated learning [EB/OL]. [2022-04-01]. https://arxiv.org/abs/1902.11175
[51] CHOI B, SOHN J Y, HAN D J, et al. Communication-computation efficient secure aggregation for federated learning [EB/OL]. [2022-04-01]. https://arxiv.org/abs/2012.05433
[52] DU W, ZENG X, YAN M, et al. Efficient federated learning via variational dropout [EB/OL]. [2022-04-01]. https://openreview.net/forum?id=BkeAf2CqY7
[53] REN J J, WANG H C, HOU T T, et al. Federated learning-based computation offloading optimization in edge computing-supported Internet of Things [J]. IEEE access, 7: 69194 – 69201. DOI: 10.1109/ACCESS.2019.2919736
[54] YU H, LIU Z L, LIU Y, et al. A fairness-aware incentive scheme for federated

learning [C]//Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society. ACM, 2020: 393 – 399. DOI: 10.1145/3375627.3375840
[55] LI T, SANJABI M, SMITH V. Fair resource allocation in federated learning [EB/OL]. [2022-04-01]. https://arxiv.org/abs/1905.10497
[56] LYU L, XU X, WANG Q, et al. Collaborative fairness in federated learning [M]//Federated Learning. Cham: Springer, 2020: 189 – 204. DOI: 10.1007/978-3-030-63076-8_14
[57] INGERMAN A, OSTROWSKI K. TensorFlow federated [EB/OL]. [2022-04-01]. https://www.tensorflow.org/federated 2019
[58] LIU Y, FAN T, CHEN T, et al. FATE: an industrial grade platform for collaborative learning with data protection [J]. Journal of machine learning research, 2021, 22(226): 1 – 6
[59] ZILLER A, TRASK A, LOPARDO A, et al. Pysyft: a library for easy federated learning [M]//Federated learning systems. Cham: Springer, 2021: 111 – 139
[60] HE C Y, LI S Z, SO J, et al. FedML: a research library and benchmark for federated machine learning [EB/OL]. [2022-04-01]. https://arxiv.org/abs/2007.13518
[61] MA Y, YU D, WU T, et al. PaddlePaddle: an open-source deep learning platform from industrial practice [J]. Frontiers of data and computing, 2019, 1(1): 105 – 115

Biographies

TANG Bo received his master's degree from Northwestern Polytechnical University, China in 2005. Currently, as the system architect of ZTE Corporation, he is mainly responsible for federated learning and AI security solutions. His current research interests include container and container cloud, heterogeneous resource scheduling, and AI security and trustworthy AI. He holds several patents in the above research areas.

ZHANG Chengming received his BS degree from Yangzhou University, China in 2011, and ME degree from Nanjing University of Posts and Telecommunications, China in 2015. He is a senior R&D engineer of ZTE Corporation. His current research interests include AI platform, container cloud, resource scheduling, federated learning, and AI security.

WANG Kewen received his BS degree from China University of Mining and Technology, China in 2015, and the ME degree from Beijing Institute of Technology, China in 2017. He is an AI senior algorithm engineer of ZTE Corporation. His current research interests include AI Systems and AI security, such as federated learning, adversarial attack, and defense in deep learning.

GAO Zhengguang received his PhD degree from Beijing University of Posts and Telecommunications, China in 2020. He was a visiting PhD student in High Performance Networks Group, University of Bristol, UK from 2018 to 2019. After graduation, he was selected for "LAN JIAN" program of ZTE Corporation as an algorithm researcher. His current research interests include 5G/6G communication technologies, mobile networks, and machine learning for future communications.

HAN Bingtao (han.bingtao@zte.com.cn) received his BS degree from Tianjin University, China in 2001, and MS degree from Nankai University, China in 2004. He is the deputy director of the State Key Laboratory of Mobile Network and Mobile Multimedia Technology, and the leader for "Adlik" project of the LF AI & Data Foundation. Currently, he is the AI system architect of Central R&D Institute, ZTE Corporation. His current research interests include deep learning algorithms, AI systems, and network intelligence. He is the author and co-author for numbers of patents and related monographs.