

Exploration across Small Silos: Federated Few-Shot Learning on Network Edge

Cong Zhao, Xinyue Sun, Shusen Yang, Xuebin Ren, Peng Zhao, Julie McCann

Abstract—Federated Learning (FL) has been drawing significant attention from both academia and industry working on distributed machine learning. In practice, learning over mutually isolated datasets residing at the network edge, also known as silos, FL clients can suffer from lack of samples, due to many reasons (e.g., expensive annotation), and this has potentially significant negative impact on FL performance. Few-Shot Learning (FSL) has been considered as a promising solution, but unfortunately cannot be directly applied to practical Cross-Silo Federated Learning (CSFL) systems. In this article, as far as we know, we conduct the first systematic discussion of the specific challenges of FSL in CSFL systems. We extract essential design issues found in Federated Few-Shot Learning (FFSL), and develop a new FFSL method based on Model-Agnostic Meta Learning (MAML). Through experiments using real-world federated datasets, we comprehensively demonstrate our method’s advantages over existing FL and FSL methods in different practical CSFL scenarios where hitherto FL and FSL methods failed. We also highlight some promising future research directions.

I. INTRODUCTION

A perfect storm of increased data availability benefiting from the proliferation of IoT and 5G technologies *etc.*, combined with much more compute availability through cloud and data-servers, enables Machine Learning (ML) to contribute in many domains (e.g., industry, finance, healthcare, and urban management). Federated Learning (FL) [1] specifically is gaining popularity as a distributed ML approach, exploiting the compute continuum via edge to cloud devices (e.g., data-centres, industrial gateways, and miniservers). It enables multiple distributed *edge clients* to collaboratively learn a global ML model under the coordination of a *cloud server*, where the raw data remain at each client. Addressing issues like network congestion, response latency, and personal privacy exposure,

This work was supported in part by the National Key Research and Development Program of China under Grant 2020YFA0713900; in part by the Alan Turing Institute; in part by the National Natural Science Foundation of China under Grant 61772410, Grant 61802298, Grant 62172329, Grant U1811461, and Grant 11690011; and in part by the China Postdoctoral Science Foundation under Grant 2020T130513 and Grant 2019M663726. (Corresponding author: Shusen Yang).

C. Zhao and J. McCann are with Department of Computing, Imperial College London (e-mails: c.zhao@imperial.ac.uk; j.mccann@imperial.ac.uk).

X. Sun is with National Engineering Laboratory for Big Data Analytics (NEL-BDA), Xi’an Jiaotong University (e-mail: sunxy2018@stu.xjtu.edu.cn).

S. Yang is with National Engineering Laboratory for Big Data Analytics (NEL-BDA) and the Ministry of Education Key Lab for Intelligent Networks and Network Security (MOE KLINNS Lab), Xi’an Jiaotong University; and also with Pazhou Laboratory (e-mail: shusenyang@mail.xjtu.edu.cn).

X. Ren and P. Zhao are with National Engineering Laboratory for Big Data Analytics (NEL-BDA) and School of Computer Science and Technology, Xi’an Jiaotong University; and also with Pazhou Laboratory (e-mails: xuebinren@mail.xjtu.edu.cn; p.zhao@mail.xjtu.edu.cn).

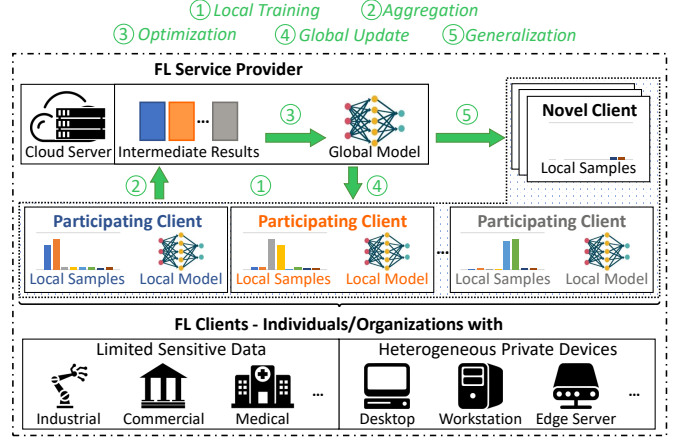


Fig. 1. The general architecture of cross-silo federated few-shot learning.

found in the predominating cloud-based centralized ML architectures, FL becomes increasingly attractive in production level ML applications [2]. However, current FL solutions find themselves inevitably bounded by issues of *non-ideal training sample distribution* in practice as distributed elements do not have a global view of the data [3].

The assumption that FL clients aim to solve ML tasks with sufficient training samples Independently and Identically Distributed (IID) across all clients does not really hold as practical clients usually encounter *ML tasks with limited well-labelled samples following unbalanced distributions* for many reasons. First, away from data-centre’s dedicated staff, accurate local sample annotation is quite expensive or impossible due to the clients’ lack of ML expertise. Further, the long-tail phenomenon, commonly observed in real-world datasets, determines that the samples extracted from different clients’ datasets usually follow highly unbalanced distributions, and the numbers of most classes of samples are severely limited. Finally, in domains where general data generators prioritize personal privacy, it is inherently difficult for clients to collect sensitive raw data (e.g., financial and medical data) and corresponding ML samples.

Illustrating this situation, we present a general FL architecture with practical clients in Fig. 1. We use a wafer defect identification task, in smart manufacturing¹ as a discussion example. Here we assume that multiple manufacturers intend to train a defect detector using FL. Due to different production environments, some manufacturers may be able to collect *relatively substantial samples from a certain number of isolated*

¹<https://www.kaggle.com/qingyi/wm811k-wafer-map>

defect classes (i.e., different manufacturers possess mutually isolated classes of samples, or *silos*), where samples in other classes are severely limited because of the long-tail phenomenon [3]. Based on general FL solutions like FedAvg [4], these manufacturers (i.e., *participating clients*) can train a *global* defect detector to accurately detect all classes of defects encountered by *all* of them. However, other manufacturers (i.e., *novel clients*) may only possess *a few samples from other unseen defect classes* due to production environment isolation and relatively limited production scale. They *cannot* directly participate in FL with insufficient samples, and the above detector *cannot* address their tasks with limited samples from unseen classes.

Therefore, it is necessary to treat the learning at the FL clients as a *Few-Shot Learning* (FSL) problem [5] that aims to achieve a model performing well not only on the tasks of participating clients, but also on the tasks with few samples from the unseen classes (i.e., FSL tasks) of novel clients. Typically, existing approaches train such a model for FSL tasks from novel clients based on *prior knowledge* in terms of data, model, and algorithm extracted from similar tasks of participating clients. Here, all clients' tasks are different but drawn from the same distribution. However, recalling that participating clients possess mutually isolated sample silos, existing methods *cannot* extract comprehensive prior knowledge for two main reasons. If the learner is deployed on the cloud server, local samples have to be uploaded, which is a breach of privacy. If the learner is deployed on an edge client, prior knowledge extracted will be biased without global generalizability. A *Federated Few-Shot Learning* (FFSL) method is therefore essential to FL.

In this article, we first present a general FL architecture across distributed sample silos in practice. A comprehensive study of FSL in the above architecture is then conducted, where the core challenges, essential to the design of FFSL methods, are extracted. After that, we develop an FFSL method based on the Model-Agnostic Meta Learning (MAML) architecture [6], whose performance was evaluated through extensive experiments based on real-world federated datasets. Finally, we discuss a set of open issues regarding the design of FFSL solutions, and then conclude this article.

II. CROSS-SILO FEDERATED LEARNING

As shown in Fig. 1, a general FL system comprises a *service provider* with a high-performance cloud server, and different numbers of *clients* possessing devices with different resource limitations (e.g., workstations, miniservers, GPU devices) at the system edge. Clients of a specific FL system are multiple individuals/organizations sharing the same learning target (e.g., an image classifier) and possess mutually isolated sample silos. The *Cross-Silo Federated Learning* (CSFL) process among the service provider and clients is as follows. For system initialization, both the service provider and all clients are deployed with the same model (with the same set of random parameters). There are **four** major steps in each training iteration of the global model, i.e.,

- *Local Training*: Each client optimizes its model with local samples, where intermediate results like gradients are generated.
- *Aggregation*: Intermediate results from all clients are aggregated by the service provider.
- *Optimization*: The service provider updates the global model by averaging aggregated intermediate results.
- *Global Update*: The updated global model is returned to each client to update its local model.

These steps are repeated until the global model converges. Note that, the global model achieved by the above approach can only address tasks of the participating clients, not that of the novel clients with limited samples from unseen classes. An FFSL method should be subtly designed for CSFL systems.

III. FEW-SHOT LEARNING IN CSFL SYSTEMS

We first review mainstream FL and FSL methods, then extract essential problems in designing an FFSL method.

A. Review on FL and FSL Methods

FL is initially introduced by Google [4]. Applications in consumer mobile services, medical record mining, financial risk prediction, smart manufacturing, *etc.* quickly emerge as a result. Considerable research into FL follows, which primarily concentrates on [1]:

- *Optimizing FL algorithms and frameworks*: Inspired by FedAvg [4], optimizations to address issues like non-IID data, resource consumption, *etc.* are discussed to enhance the FL efficiency [7].
- *Enhancing FL privacy preservation*: Except for the local sample residence, advanced methods like secure multi-party computation, homomorphic encryption, differential privacy, *etc.* are used to further prevent adversaries from inferring privacy based on FL intermediate results [8].
- *Constructing FL platforms*: Both academic and commercialized FL platforms like TensorFlow Federated (TFF), Federated AI Technology Enabler (FATE), PySyft, *etc.* emerge and stimulate platform refinement and application development.

However, existing studies neither explicitly discuss nor effectively address *tasks with limited samples*, which is a common issue for real-world clients [3].

FSL, a sub-area ML problem of learning from limited samples, quickly rises and promotes applications in computer vision, natural language processing, robotics, *etc.* Existing FSL methods generally use *prior knowledge* to enhance following for rapid generalization on new tasks with few samples [5]:

- *Data*: These methods increase the sample number by augmenting limited samples through either hand-crafted rules determined by domain experts or advanced data-driven models (e.g., Generative Adversarial Networks) learnt from auxiliary data.
- *Model*: These methods constrain the space containing the optimal hypothesis of the targeted model to enable a good approximation with limited samples. Here, the space is reduced through models or structures (e.g., embedding) learnt from auxiliary tasks.

- *Algorithm*: These methods optimize the model approximation process through either a good model initialization or a more targeted searching strategy (e.g., meta-learned model optimizer) learnt from auxiliary tasks.

Existing approaches depend on different forms of *prior knowledge* centrally learnt from auxiliary data and tasks. The *full access to different auxiliary tasks*, however, *cannot* be directly achieved in CSFL systems.

B. Essential Design Problems

Federated Few-Shot Learning (FFSL) is critical to real-world applications of FL. There exists no systematic discussion about FSL in CSFL systems, and related research is quite limited. In [9] and [10], Few-Shot Federated Learning is investigated to achieve FL with only a few global model update iterations, which, essentially different from FFSL, aims to address edge-cloud communication reduction instead of tasks with limited unseen samples. In [11], FedAvg [4] is interpreted as a meta learning algorithm, and a few-shot fine-tuning stage is integrated, where the initial global model trained by FedAvg is trained with FSL tasks drawn from all participating clients. This enables the trained global model to achieve a higher accuracy on different participating clients by further optimizing the deployed model with only a few local samples, which, unlike FFSL, does not directly address tasks with limited samples from unseen classes. In [12] and [13], FFSL is initially explored based on algorithm- (*i.e.*, MAML) and model-enhancement (*i.e.*, RelationNet), respectively. Here, FSL tasks are extracted from participating clients to train a global model that can address unseen FSL tasks of novel clients. However, unaligned to our target, these methods are neither specifically designed for, nor comprehensively validated in, CSFL scenarios (*i.e.*, all clients possess mutually isolated classes of samples). Besides, they introduce either intense computation (*i.e.*, adversarial learning in [12]) or a heavy few-shot learner (*i.e.*, the deep K-tuplet network in [13]) to clients, which may not be applicable to edge devices with strict resource limitations.

To design an effective FFSL method, following key problems should be addressed simultaneously:

- *Non-IID silos*: Samples drawn from different real-world clients' local silos could be non-IID in many ways. Effects like label and feature distribution skews, quantity skews, *etc.* should be fully considered.
- *Model generalization and personalization*: In CSFL systems, the trained model should be applicable to both participating clients and novel clients. Participating clients should be able to personalize the model with few local samples, and novel clients should get a model that can address local tasks with limited unseen samples.
- *Clients with limited resources*: For different clients, resources in terms of computation, communication, *etc.* for FL are different and usually strictly limited. The FFSL method and model should be as lightweight as possible to avoid client incompetence and encourage client participation with 'inexpensive learning'.

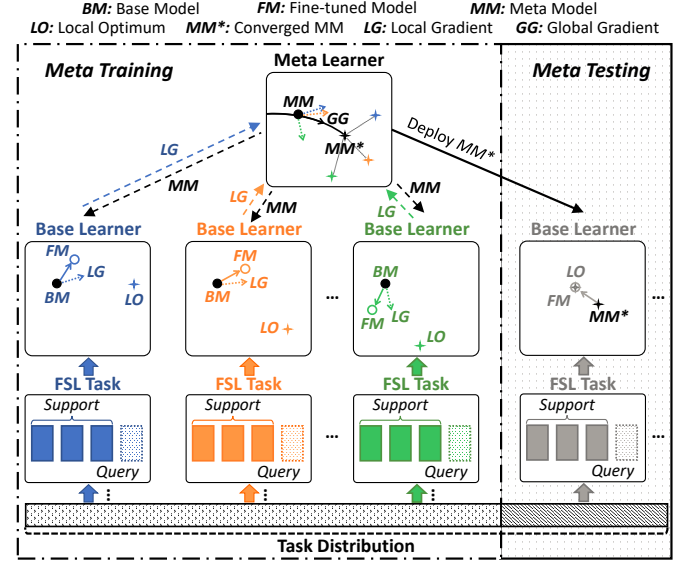


Fig. 2. FSL based on MAML.

IV. FEDERATED FEW-SHOT LEARNING BASED ON MODEL-AGNOSTIC META LEARNING

We develop an FFSL method based on *Model-Agnostic Meta Learning* (MAML) [6]. We first briefly introduce MAML, and elaborate its advantages on addressing the FSL problem in CSFL systems. Then, our method is presented.

A. Introduction to MAML

MAML is an algorithm-enhancing FSL method, learning an *optimal model initialization* for different FSL tasks drawn from a same distribution. The learnt model can adapt to *novel tasks* from the same distribution with only a *few samples*. MAML imposes *no restrictions* on the form of model, as long as it can be optimized through gradient decent. The general workflow of MAML is illustrated in Fig. 2.

MAML comprises two layers of models (or *learners*), *i.e.*, the *meta learner* and multiple *base learners*. Initially, both the meta learner and all base learners are deployed with the same model with the same set of random parameters. For one update iteration of the meta model in meta training, all base learners independently draw different FSL tasks from a task distribution (random tasks with samples from isolated classes are drawn priorly and held out as *novel tasks* for meta testing). Here, each FSL task contains limited *support samples* (*i.e.*, few-shots) and *query samples* drawn from the same set of classes. Instead of directly descending towards the potential optimal model for the task (*i.e.*, the local optimum), each base learner first optimizes the base model through *one step of vanilla gradient descent* using the support samples (*i.e.*, model *fine-tuning*), and then determines a descending decision that minimizes the *fine-tuned model's* loss on the query samples (*i.e.*, the local gradient). The meta learner then updates the meta model through *one step of Adam-based gradient descent*, where the global gradient is computed by averaging all local gradients. The updated meta model is replied to all base learners as the updated based model. Such an iteration is

repeated until the meta model converges. The trained meta model can be treated as effective only when it performs well on held out novel tasks (*i.e.*, achieves a high accuracy on the query samples *after* fine-tuned by the support samples in each novel task). In fact, the meta learner takes different *FSL tasks* as meta training ‘*samples*’, where the meta model descends along the direction that is *most sensitive to few-shot fine-tuning* (*i.e.*, with the highest accuracy gain).

MAML inherently fits FL due to its dual-layer architecture enabling local residence of raw data. Compared to other methods, MAML has more potentials in CSFL systems:

- *Resilience to non-IID silos*: FL across mono-targeted clients with non-IID silos is comparable to MAML on different tasks drawn from the same distribution. For CSFL clients, the ‘*non-IIDness*’ of FSL tasks over the task distribution is inherently *lower* than that of base samples over the sample distribution, which induces less impacts on the meta model training.
- *Model adaptability*: MAML’s output is the optimal model initialization (*i.e.*, the meta model) for different (sets of) FSL tasks over the same distribution. Fine-tuned with few samples, the model can be personalized for tasks of participating clients, and can effectively address tasks with limited samples from unseen classes of novel clients.
- *Lightweight*: MAML introduces no extra model architecture apart from the targeted model shared by the meta learner and all base learners. Meanwhile, introducing no extra computation, MAML’s resource friendly model update using few samples is applicable to clients with strict resource limitations.

B. The MAML-based FFSL Method

We develop an FFSL method based on MAML for CSFL systems with the following workflow.

1) *FL Task Publication and System Initialization*: For the CSFL system in Fig. 1, the service provider publishes FL tasks according to market research or client commission. For a specific FL task, interested individuals/organizations register at the service provider as FL clients. Particularly, as described in Section I, those with silos containing relatively substantial samples are registered as *participating clients*, who contribute to the federated training of a global FSL model. Others with silos containing limited samples are registered as *novel clients*, who do not participate in federated model training due to sample deficiency, but directly use the global model trained by participating clients to address their own novel tasks with limited samples from unseen classes (*i.e.*, classes of samples that are not used for federated model training). In this case, when all participating clients are determined, an initial model is deployed at the cloud server as the *meta learner* in Fig. 2 (*i.e.*, the global FSL model). Each edge participating client is deployed with the same model as a *base learner* in Fig. 2.

2) *Federated Training of the Meta Model*: The meta model is optimized iteratively during the federated training process. Steps for each meta model update iteration, where corresponding operations are similar to that in Subsection IV-A, are:

- **Step 1**: All participating clients separately draw *one* different FSL task from their local silos with isolated

classes of samples. For example, for a 2-way-1-shot task, one sample from each of the two local classes is selected, regarded as two *support samples*. Similarly, two *query samples* (different from the support samples) are selected. Note that, for each participating client, FSL tasks can be extracted either in advance to form a local task set, or on-the-fly from incoming local streams.

- **Step 2**: Each participating client fine-tunes its base model using all support samples. Then, with the base model before fine-tuning and the fine-tuned model’s loss on all query samples, each client computes its local gradient and sends it to the service provider.
- **Step 3**: The service provider computes the global gradient by uniformly averaging all local gradients, and updates the meta model through one step of Adam-based gradient descent. The updated meta model is replied to all participating clients.
- **Step 4**: Each participating client updates its base model as the received meta model.

The steps above are repeated until the meta model converges. Note that, in Step 1, each client uses *one* FSL task for base model fine-tuning to eliminate the impact of the unbalanced distribution of available FSL tasks on different clients.

3) *Meta Model Personalization and Generalization*: For each participating client, the converged meta model can be further fine-tuned with new samples extracted from the client’s local silo for personalization. For novel clients (*i.e.*, individuals/organizations with few samples from unseen classes), they can directly use the converged meta model to address their own FSL tasks (*i.e.*, novel meta testing FSL tasks).

V. EVALUATION

We evaluate the performance of our method under different CSFL settings.

A. Methodology

We implemented our FFSL method with PyTorch. Using Docker containers, we constructed a CSFL simulator containing one cloud server (a container without raw data acting as the meta learner), and a variable number of clients (different containers, each with a local silo and acts as the base learner on a client). We compared the performance of FFSL with that of typical FL (*i.e.*, FedAvg [4]) and FSL (*i.e.*, MAML [6]) methods in different CSFL scenarios. Specifically, FedAvg was deployed across the cloud server and all clients. MAML, which was not designed for FL however, was locally deployed on a random subset of clients to preserve privacy.

B. CSFL Cases, Datasets, and Models

As illustrated in Fig. 3, we considered **three** different CSFL cases, *i.e.*, *Hand-written Digit Classification* (HDC) for mobile service providers, *Wafer Defect Identification* (WDI) for smart manufacturers, and *Engine Degradation Prediction* (EDP) for asset predictive maintainers.

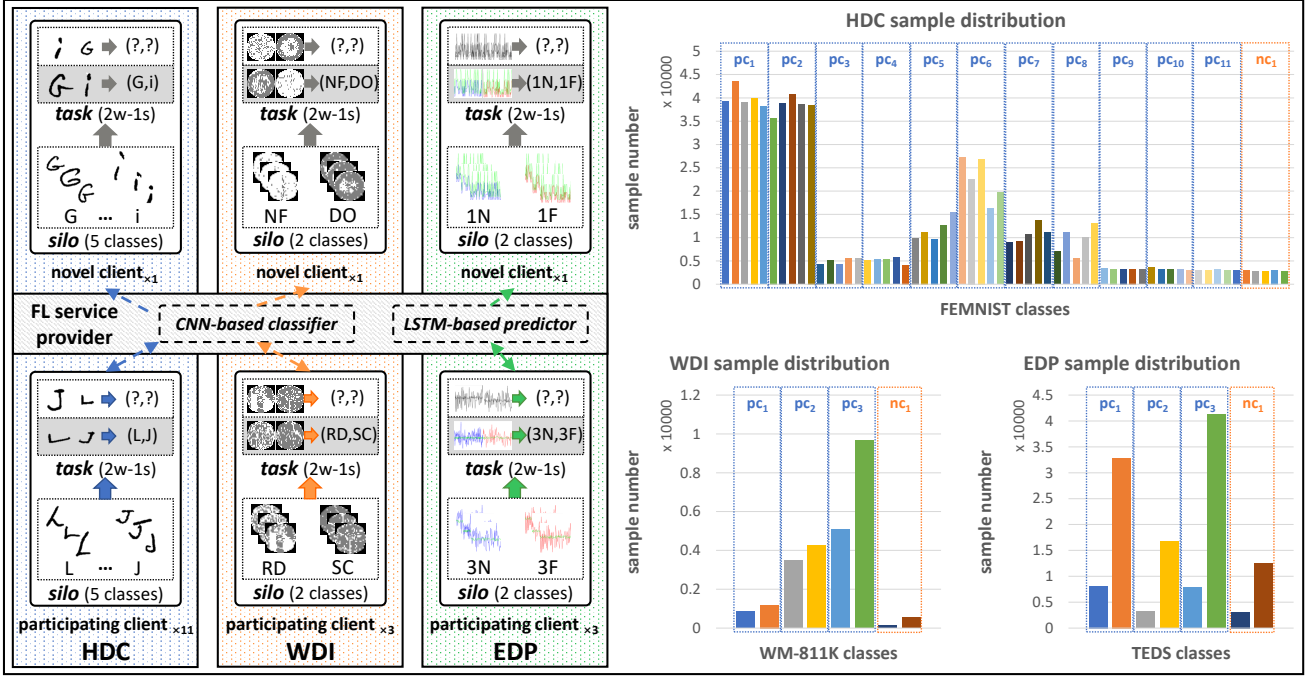


Fig. 3. Experimental CSFL scenarios.

HDC: We used the FEMNIST dataset² containing 62 non-IID classes of hand-written digits for HDC. Each of 11 participating clients had five exclusive classes of digits. Remained classes were treated as data at novel clients. A CNN-based 2-way-1-shot classifier³ was trained. Note that, as mentioned in Section IV, a 2-way-1-shot task contained two support samples and two query samples from two different classes, respectively.

WDI: We used the WM-811K dataset⁴ containing eight non-IID classes of wafer maps with different defects for WDI. Each of three participating clients had two exclusive classes of wafer maps. Remained classes were treated as data at novel clients. A 2-way-1-shot classifier with the same structure of HDC’s model was trained.

EDP: We used the Turbofan Engine Degradation Simulation (TEDS) dataset⁵ for EDP, and the dataset contained non-IID sensing measurement time-series collected from engines deployed under four different working conditions. Each of three participating clients had series collected under an exclusive working condition. Remained data were treated as data at novel clients. In EDP, we considered the 50-to-30 binary prediction problem, *i.e.*, for a series containing measurements collected within 80 continuous time slots, using the first 50 slots to predict whether the engine will fail or not in the next 30 slots. Series at each client were organized as two classes of samples labelled with ‘normal’ and ‘failure’, respectively. An LSTM-based 2-way-1-shot predictor⁶ was trained.

We conducted two groups of experiments with an *unbal-*

anced and *balanced* number of samples at each participating client, respectively. For the unbalanced group the detailed sample distribution is illustrated in Fig. 3. For the balanced group, with the class isolation setting in Fig. 3, the same number of samples (*i.e.*, the sample number of the class with the least samples among all classes at all participating clients) from each class were used by each participating client. For a fair comparison, when the model was trained using different methods, each participating client used *one* 2-way-1-shot task randomly drawn from its local silo in each global model update iteration. MAML and FFSL trained the model according to procedures in Subsections IV-A and IV-B, respectively, where the learning rate of base model fine-tuning was 0.4, and the initial learning rate of Adam-based meta model update was 0.001. For FedAvg, in each global model update iteration, each client used all support and query samples in the FSL task to update its local model through two steps of Adam-based gradient descent with an initial learning rate of 0.001, then the global model was updated by uniformly averaging all updated local models.

C. Results

In HDC, WDI, and EDP, we periodically evaluated the performance of the model iteratively optimized by FFSL, FedAvg, and MAML. For each round of evaluation, 100 random FSL tasks were drawn from participating clients and novel clients, and were fed to the model trained by different methods, respectively. As shown in Fig. 4, corresponding *average training accuracy* on FSL tasks of participating clients (*e.g.*, *ffsl_tr*) and *average testing accuracy* on FSL tasks of novel clients (*e.g.*, *ffsl_te*) were recorded to demonstrate the model’s personalization and generalization capabilities. Note that, to achieve the training accuracy on each FSL task of

²<https://github.com/TalwalkarLab/leaf/tree/master/data/femnist>

³<https://github.com/cbfinn/maml>

⁴<https://www.kaggle.com/qingyi/wm811k-wafer-map>

⁵<https://ti.arc.nasa.gov/tech/dash/groups/pcoe/prognostic-data-repository/#turbofan>

⁶<https://github.com/umbertogriffo/Predictive-Maintenance-using-LSTM>

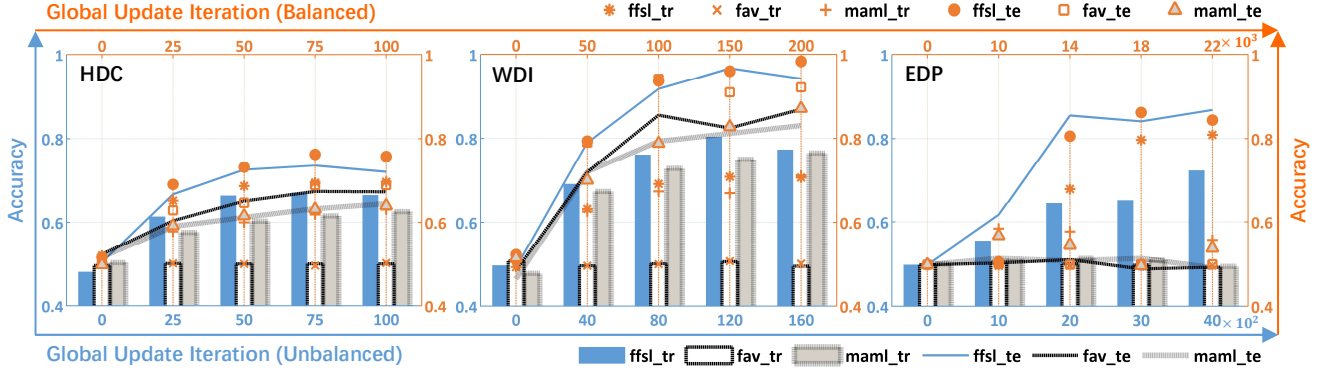


Fig. 4. Performance of FFSL, FedAvg, and MAML in different CSFL scenarios.

participating clients, the model trained by FFSL and MAML was fine-tuned with the support samples then tested on the query samples, and that trained by FedAvg was tested on all support and query samples. To achieve the testing accuracy on each FSL task of novel clients, the model trained by all three methods was fine-tuned with the support samples then tested on the query samples to get the testing accuracy.

According to the results, in all scenarios, the model trained by FFSL manages to converge across silos at all participating clients (*i.e.*, in HDC/WDI/EDP, the training accuracy achieves 67%/77%/72% after 100/160/4000 global model update iterations with the unbalanced setting, and 70%/71%/81% after 100/200/22000 iterations with the balanced setting). Simultaneously, the model manages to generalize on silos at novel clients (*i.e.*, in HDC/WDI/EDP, the testing accuracy achieves 72%/94%/87% with the unbalanced setting, and 76%/98%/84% with the balanced setting). Note that, under the same fine-tuning setting, the model would achieve a higher accuracy on client silos whose optimal hypothesis is closer to it. Due to the silo non-IIDness, different clients may have different model-hypothesis distances in our experiments. Since the single novel client's model-hypothesis distance is shorter than the average distance of different participating clients, the testing accuracy is higher than the training accuracy. It is obvious that *FFSL outperforms FedAvg and MAML in terms of both training accuracy and testing accuracy* under all CSFL settings. We believe this is due to FedAvg and MAML's incompatibility to either the *FSL setting* or the *non-IID silos*.

Impact of the FSL Setting: Designed as FSL methods, both FFSL and MAML manage to converge on silos at participating clients, except for MAML in EDP (*i.e.*, achieving a training accuracy no more than 56%). We believe that the LSTM model used for EDP cannot be effectively trained based on a single local silo with limited tasks of any specific participating client, where, as demonstrated by FFSL, FL is necessary. As an FL method without FSL consideration, FedAvg is far from converged in all scenarios (*i.e.*, achieving a training accuracy no more than 51%). Treating samples in FSL tasks as conventional mini-batches, it requires significantly more global model update iterations before the potential convergence. Interestingly, FedAvg achieves a high testing accuracy in HDC and WDI for two reasons. First, model fine-tuning

with support samples in the FSL task of novel clients surely improves the model's accuracy on query samples. Then, as the training proceeds, FedAvg descends (slowly though) towards the optimal hypothesis for classifying all classes of samples of all participating clients. This induces gradually increasing testing accuracy if the optimal hypothesis for classifying unseen classes of novel clients is close to the above hypothesis of participating clients, which, however, is silo-specific without generality. Since FedAvg is not designed for FSL, the testing accuracy of the fine-tuned model is lower than that of FFSL.

Impact of Non-IID Silos: According to Fig. 3 and Fig. 4, with the same class isolation setting, since that, according to our design, each participating client only uses one FSL task in each global model update iteration, the converging property of FFSL, FedAvg and MAML remains almost the same in scenarios with different sample distributions (*i.e.*, comparison between the unbalanced and balanced group of HDC, WDI, and EDP, respectively). Then, due to the class isolation setting, MAML, as an FSL method without FL consideration, achieves a lower testing accuracy compared to FFSL in all scenarios. The reason for this is that the model trained through MAML is optimized on a local silo only, where the global prior knowledge cannot be extracted. Therefore, the trained model cannot effectively generalize on silos of novel clients.

VI. OPEN RESEARCH ISSUES

Discussions above highlight challenges of FSL in CSFL, however there are more issues worth discussing further.

General FFSL based on Multi-Task Learning: Our method manages to learn a model addressing FL tasks with different targets simultaneously (*e.g.*, HDC and WDI). This can help FL service providers to attract more clients and reduce the training cost. Introducing multi-task learning will also alleviate sample deficiency. The *similarity model* of FL tasks with different targets is challenging but necessary to construct.

Online FFSL with Dynamic Clients: For CSFL, clients may quit and join the federated training process for their own benefits. It is promising to treat FFSL as an online learning process upon FSL task streams. Core issues like *optimization complexity* and frequent *concept shift* need to be addressed.

FFSL Client Incentivization: Due to the lack of samples, clients may choose to join FFSL tasks in an intermittent

manner, where an incentive method can stimulate active participations. For the service provider, participating clients should be rewarded according to their contributions. An essential issue is the *quantification* of client contribution, especially in terms of *samples* considering the FSL setting.

VII. CONCLUSION

In this article, we conducted, we believe, the first systematic discussion of the few-shot learning problem with regard to practical cross-silo federated learning systems implemented on the network edge, and explicitly quantified the severity of the impact that tasks with insufficient samples have on the system and which cannot be eliminated using existing methods. Concentrating on the deficiency of existing federated learning and few-shot learning solutions, we extracted the essential design considerations and presented a federated few-shot learning method based on MAML. Using three different validating cases, considering mobile service, smart manufacturing, and predictive maintenance, our method achieved effective federated few-shot learning with good model personalization and generalization capabilities where prior federated learning and few-shot learning methods failed due to their incompetence in addressing tasks with limited samples and mutually exclusive edge data silos, respectively. We finally discussed promising directions of federated few-shot learning research, where the introduction of multi-task learning, online learning, and incentivization should be further explored.

REFERENCES

- [1] P. Kairouz *et al.*, “Advances and open problems in federated learning,” Mar. 2021; <https://arxiv.org/abs/1912.04977v3>, accessed August 11, 2021.
- [2] K. Yang *et al.*, “Federated machine learning for intelligent iot via reconfigurable intelligent surface,” *IEEE Netw.*, vol. 34, no. 5, 2020, pp. 16–22.
- [3] J. Shu, Z. Xu, and D. Meng, “Small sample learning in big data era,” Aug. 2018; <https://arxiv.org/abs/1808.04572v3>, accessed August 11, 2021.
- [4] B. McMahan *et al.*, “Communication-efficient learning of deep networks from decentralized data,” in *Proc. of PMLR AISTATS*, 2017, pp. 1273–1282.
- [5] Y. Wang *et al.*, “Generalizing from a few examples: A survey on few-shot learning,” *ACM CSUR*, vol. 53, no. 3, 2020, pp. 63:1–63:34.
- [6] C. Finn, P. Abbeel, and S. Levine, “Model-agnostic meta-learning for fast adaptation of deep networks,” in *Proc. of PMLR ICML*, 2017, pp. 1126–1135.
- [7] R. Yu and P. Li, “Toward resource-efficient federated learning in mobile edge computing,” *IEEE Netw.*, vol. 35, no. 1, 2021, pp. 148–155.
- [8] C. Ma *et al.*, “On safeguarding privacy and security in the framework of federated learning,” *IEEE Netw.*, vol. 34, no. 4, 2020, pp. 242–248.
- [9] A. Kasturi, A. R. Ellore, and C. Hota, “Fusion learning: A one shot federated learning,” in *Proc. of Springer ICCS*, 2020, pp. 424–436.
- [10] N. Guha, A. Talwalkar, and V. Smith, “One-shot federated learning,” Mar. 2019; <https://arxiv.org/abs/1902.11175v2>, accessed August 11, 2021.
- [11] Y. Jiang *et al.*, “Improving federated learning personalization via model agnostic meta learning,” Sep. 2019; <https://arxiv.org/abs/1909.12488v1>, accessed August 11, 2021.
- [12] C. Fan and J. Huang, “Federated few-shot learning with adversarial learning,” Apr. 2021; <https://arxiv.org/abs/2104.00365v1>, accessed August 11, 2021.
- [13] W. Zheng *et al.*, “Federated meta-learning for fraudulent credit card detection,” in *Proc. of IJCAI*, 2020, pp. 4654–4660.

BIOGRAPHY

Cong Zhao (c.zhao@imperial.ac.uk) received his Ph.D. degree in Computer Science and Technology from Xi’an Jiaotong University (XJTU) in 2017. He is currently a research associate in the Department of Computing at Imperial College London. His research interests include meta learning, federated learning, and industrial intelligence.

Xinyue Sun (sunxy2018@stu.xjtu.edu.cn) received her B.Sc. degree from Northeastern University in 2018. She is currently working towards her Ph.D. degree in Xi’an Jiaotong University. Her research interests include small sample learning and edge intelligence.

Shusen Yang (shusenyang@mail.xjtu.edu.cn) is a Professor and Director of NEL-BDA and Deputy Director of MOE KLINNS Lab at XJTU, and the Deputy Director of IAIC at Pazhou Laboratory. He is a senior member of IEEE. His research focuses on distributed systems and data sciences, and their applications in industrial scenarios.

Xuebin Ren (xuebinren@mail.xjtu.edu.cn) received his Ph.D. degree from Xian Jiaotong University, China, in 2017. He is currently an Associate Professor with Xian Jiaotong University. He has been a visiting Ph.D. student at Imperial College London, UK, from 2016 to 2017. His research interests focus on data privacy protection, federated learning.

Peng Zhao (p.zhao@mail.xjtu.edu.cn) received the B.S. and Ph.D. degrees in computer science and technology from Xi’an Jiaotong University, China, in 2007 and 2013 respectively. He is currently an Associate Professor with Xi’an Jiaotong University, China. His research interests include distributed systems, cloud computing, and system optimization.

Julie McCann (j.mccann@imperial.ac.uk) is a Professor of Computer Systems; leader of AESE; Directs pan-Imperial SCFC; PI Digital Oceans, Eco-Cities; co-directed ICRI-SC. Her research interests: decentralized self-organizing IoTs/CPSs and scalable algorithms for spatial computing systems; low-powered smart dust. She is an elected peer for EPSRC and a Fellow of the BCS.