Bulletin of the Technical Committee on

Data Engineering

March 2021 Vol. 44 No. 1

IEEE Computer Society

Letters

Special Issue on Learning and Reasoning on Knowledge Graphs and Applications

Conference and Journal Notices

The TC on Data Engineering

Membership in the TC on Data Engineering is open to all current members of the IEEE Computer Society who are interested in database systems. The TCDE web page is http://tab.computer.org/tcde/index.html.

The Data Engineering Bulletin

The Bulletin of the Technical Committee on Data Engineering is published quarterly and is distributed to all TC members. Its scope includes the design, implementation, modelling, theory and application of database systems and their technology.

Letters, conference information, and news should be sent to the Editor-in-Chief. Papers for each issue are solicited by and should be sent to the Associate Editor responsible for the issue.

Opinions expressed in contributions are those of the authors and do not necessarily reflect the positions of the TC on Data Engineering, the IEEE Computer Society, or the authors' organizations.

The Data Engineering Bulletin web site is at
http://tab.computer.org/tcde/bull_about.html.

Letter from the Editor-in-Chief

The March issue of the Data Engineering Bulletin focuses on the intricate interplay as well as a significant gap between data management and machine learning when it comes to supporting real-life business applications.

The opinion piece of this issue features a group of distinguished researchers and their assessment and prognosis of machine learning's current and future roles in building database systems. Besides highlighting several specific potentials and challenges such as using machine learning to optimize database indices and query optimization, the article also gives a great overview of how databases, data analytics and machine learning, system and infrastructure, work together to support today's business needs. It is clear that the business needs, the volume, velocity, and variety of the data, the latency and throughput requirements have evolved dramatically and in consequence, data management systems must adapt. The opinion pieces described four disruptive forces underneath the evolution, which are likely to influence future data systems.

Our associate editor Sebastian Schelter put together the current issue–Data Validation for Machine Learning Models and Applications–that consists of six papers from leading researchers in industry and academia. The papers focus on data validation, which is a critical component in end-to-end machine learning pipelines that many business applications rely on.

Haixun Wang
Instacart

Letter from the Special Issue Editor

Federated learning has attracted much attention recently, as machine learning has been widely applied to many real applications while security concern of data and model has also arisen. In distributed systems, when the server should have interactions with clients, users might prefer the server being not able to access each client's personal raw data. When different data owners want to share insight into their data, they may also share only models instead of raw data. Such practical usage scenarios triggered many interesting new designs of federated learning algorithms and systems, which can be summarized as horizontal federated learning, vertical federated learning, and federated transfer learning. In this issue, we included several papers covering different perspectives of federated learning which makes it a very interesting one pointing out several potential new directions of the field.

The first paper *Federated Computing: Query, Learning, and Beyond* and the second paper *Federated Learning without Full Labels: A Survey* surveyed existing federated computing strategies and discussed the future trends of the field. The following papers *FedCLIP: Fast Generalization and Personalization for CLIP in Federated Learning*, *Reconciling Security and Communication Efficiency in Federated Learning*, and *Accelerated Federated Optimization with Quantization* studied potential new approaches to improve the effectiveness and efficiency of federated learning algorithms. Then, *Federated Truth Discovery for Mobile Crowdsensing with Privacy-Preserving Trustworthiness Assessment*, *Federated Ensemble Learning: Increasing the Capacity of Label Private Recommendation Systems*, and *Enhance Mono-modal Sentiment Classification with Federated Cross-modal Transfer* discussed the applications to mobile crowdsensing, recommender systems, and sentiment analysis. Last but not least, the paper *NVIDIA FLARE: Federated Learning from Simulation to Real-World* from NVIDIA has introduced their systematic development of an open-source software development kit that can benefit the whole field for research study and real application development.

I would like to thank all the authors for their contributions, making this issue a significant and interesting discussion about present and future research directions related to federated learning.

Yangqiu Song
The Hong Kong University of Science and Technology

Federated Computing: Query, Learning, and Beyond

Yongxin Tong[†] Yuxiang Zeng[†,‡] Zimu Zhou[♯] Boyi Liu[†] Yexuan Shi[†]
Shuyuan Li[†] Ke Xu[†] Weifeng Lv[†]
[†] State Key Laboratory of Software Development Environment,
Beijing Advanced Innovation Center for Future Blockchain and Privacy Computing,
School of Computer Science, Beihang University, Beijing, China
{yxtong,turf1013,liuby,skyxuan,lishuyuan,kexu,lwf}@buaa.edu.cn
[‡] The Hong Kong University of Science and Technology, Hong Kong SAR, China
[♯] City University of Hong Kong, Hong Kong SAR, China zimuzhou@cityu.edu.hk

Abstract

Big data has played an important role in the development of the economy. However, the "data isolation" problem largely hinders its full potential. To solve this problem, it is crucial to enable collaborative computing among multiple data sources. Federated computing, a new collaborative computing paradigm that keeps the raw datasets decentralized, has emerged as a hot research topic in both databases and artificial intelligence. This paper introduces the concepts of federated computing, reviews recent topics, which include "federated queries" and "federated learning", and discusses the future trends.

1 Introduction

In the era of big data, governments, organizations, companies, and individuals rely heavily on data analysis for decision-making. However, the "data isolation" problem remains a major bottleneck for big data analysis. As the name implies, data is stored as islands among multiple data owners, which seriously hinders the sharing and circulation of big data. For example, medical big data can be used to build accurate machine learning models for disease prediction and diagnosis. However, patients' data, *e.g.*, medical examination results, may distribute across multiple autonomous hospitals, which hinders joint analysis due to privacy concerns. Another example is the smart city applications with big spatiotemporal data. In these applications, the spatiotemporal data are distributed in multiple platforms, *e.g.*, taxi and ride data on multiple travel platforms, meal data on the catering platform, and cell tower data on telecommunications operation service providers. Due to the privacy concerns, it is difficult to collect these data directly, which hinders the rapid response of smart city applications. Therefore, how to break the data isolation and unite multiple data owners for computation is crucial to strengthen the circulation of data and the development of the big data industry.

To break the data isolation dilemma, "federated computing" is proposed as a new collaborative computing paradigm, where multiple data owners are united as a data federation and process collaborative computing while keeping the raw data locally at each data owner [1, 2, 3]. Its idea is to push the computation on raw data to the

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

data owners, while only summary information from the raw data is communicated back to the server for secure aggregation.

Challenges. Federated computing mainly faces three challenges. *(i) Privacy.* Federated computing poses new privacy constraint, *i.e.*, the raw data of each data owner is kept locally, while allowing collaborative computing among data owners. *(ii) Efficiency.* Since privacy and security constraints often introduce extra communication and computation, it is important to comply with these constraints while allowing high-efficiency computing. *(iii) Effectiveness.* As the data of multiple data owners may not be independent and identically distributed (non-IID), how to conduct accurate analysis in presence of data heterogeneity is also a challenge.

Milestones. Two hot topics in current federated computing are "federated queries" and "federated learning".

- The federated queries indicate that the data federation should support secure queries over multi-party data owners. It serves as the basis of federated computing. Several data federation systems have been proposed to support various federated queries, such as SMCQL [2], Hu-Fu [4], FedGraph [5], which are designed for relational, spatial, and graph queries, respectively.
- Upon a data federation, machine learning algorithms can be further implemented to support upper-layer applications. Such algorithms are often known as "federated learning". For example, FedAvg [6] is one of the most popular federated learning algorithms, and there have been extensions to more generic frameworks [1].

Roadmap. In the rest of this paper, we first introduce the basic concepts and general framework of federated computing in Section 2. Then, we review the related work on federated queries (Section 4.2) and federated learning (Section 4), respectively. Finally, we summarize the future directions in Section 5 and conclude in Section 6.

2 Basic Concepts and General Framework

This section introduces the basic concepts and general framework of federated computing in Section 2.1 and Section 2.2, respectively.

2.1 Basic Concepts of Federated Computing

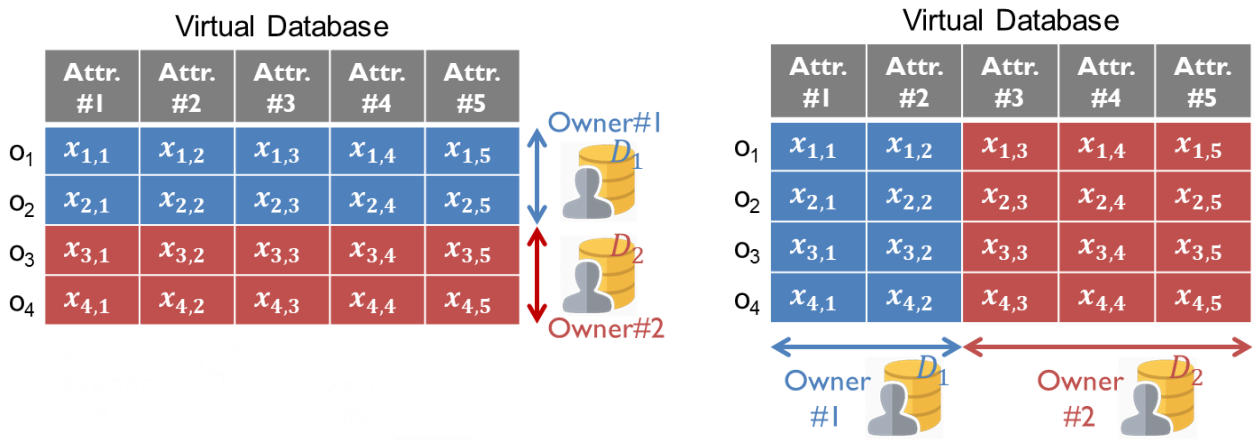
In the following, we introduce the key concepts in federated computing and the problem statement.

Data Federation. Federated computing is performed over a special data system, *i.e.*, *data federation*.

Definition 1 (Data Federation): A data federation $\mathcal{F}$ is a federation of local datasets $\{\mathcal{D}_1, \mathcal{D}_2, \cdots, \mathcal{D}_m\}$ held by m data owners, where each local dataset $\mathcal{D}_i$ has n_i objects $\{o_1, o_2, \cdots, o_{n_i}\}$ and each object o_j has k_j attributes $\{x_1, x_2, \cdots, x_{k_j}\}$. Then, the (virtual) database of this data federation is denoted by the union of these local datasets, *i.e.*, $\mathcal{D} = \mathcal{D}_1 \cup \mathcal{D}_2 \cup \cdots \cup \mathcal{D}_m$.

When local datasets are data silos of data owners like enterprises and organizations, we call this setting "cross-silo". By contrast, when data owners are individuals like mobile phone users or edge device users, we call this setting "cross-device". Accordingly, the application settings of federated computing can be categorized into *cross-silo* and *cross-device* [7]. For example, a data federation system was built to answer collaborative queries over clinical data across organizations under the cross-silo setting [8], while Google allowed their mobile users to jointly train language models under the cross-device setting [6].

Data Integration Mode. On the other hand, from the perspective of how the local datasets are integrated into the virtual database, a data federation can be categorized into *horizontal data federation* (*a.k.a.*, horizontally



(a) Horizontal data federation

(b) Vertical data federation

Figure 1: Two ways for data owners' local datasets to integrate into a data federation, including (a) horizontal data federation and (b) vertical data federation.

partitioned data [9]) and *vertical data federation* (*a.k.a.*, vertically partitioned data [10]). As shown in Figure 1, in a horizontal data federation, each data owner has a disjoint subset of rows in the virtual database. By contrast, in a vertical data federation, each data owner has a disjoint subset of columns in the virtual database. Both types are commonly seen in existing literature [1].

Problem Statement. A formal definition of federated computing is as follows.

Definition 2 (Federated Computing): Given a data federation $\mathcal{F}$ of m data owners $\{\mathcal{D}_i\}$ and a computing task $\mathcal{T}(\mathcal{D})$ over the (virtual) database $\mathcal{D} = \cup_{i=1}^{m} \mathcal{D}_i$, a federated computing algorithm aims to compute the result of $\mathcal{T}(\mathcal{D})$ while satisfying the following constraints:

- **Autonomous Constraint**: any data owner does not share his raw data to others.
- **Security Constraint**: during the computation, except for the result, any sensitive data of a data owner cannot be leaked to others.

In Definition 2, the autonomous constraint is aligned with real-world scenarios, since a data owner would like to autonomously manage his local dataset.

Attacker Models. The security constraint is due to the concern of privacy attacks, which are commonly seen nowadays and can be categorized into two kinds, *semi-honest adversary* and *malicious adversary*.

- **Semi-honest Adversary**: a semi-honest adversary will honestly follow the specified computation procedure, but may attempt to infer sensitive data of data owners in the meantime.
- **Malicious Adversary**: a malicious adversary may arbitrarily deviate from the specified computation procedure to infer sensitive data.

2.2 General Framework for Federated Computing

Solutions to federated computing can be summarized into a two-phase general framework. The *main idea* of this general framework is as follows.

- **Local Computation.** First, motivated by the idea of computation pushdown, a federated computing task is decomposed into several computing tasks over the local datasets, and hence data owners can locally compute their partial results.
- **Secure Computation.** After that, a well-designed protocol is performed across data owners to derive the final result based on their partial results in a privacy-preserving manner. Finally, the final answer will be returned to the service user.

The federated computing task $\mathcal{T}$ in existing work can be mainly classified into two kinds, *federated query* and *federated learning*, which will be elaborated later in Section 4.2 and Section 4, respectively. To evaluate the performance of solutions to federated queries and federated learning, commonly-used metrics include result accuracy, time efficiency, communication cost, etc.

3 Federated Queries

In this section, we introduce how to process federated queries over a data federation. Specifically, we first present the key concepts and main workflow of federated queries in Section 3.1. Then, we introduce the techniques of processing federated queries from three aspects, query rewrite (Section 3.2), local queries (Section 3.3), and secure operations (Section 3.4). Finally, we summarize these studies in Section 3.5.

3.1 Overview

Problem Statement. Based on Definition 3, we introduce the formal definition of federated queries as follows.

Definition 3 (Federated Query): Given a data federation $\mathcal{F}$ of m data owners $\{\mathcal{D}_i\}$ and a query request $\mathcal{Q}$ over the (virtual) database $\mathcal{D} = \cup_{i=1}^{m} \mathcal{D}_i$, a federated query aims to retrieve the result $\mathcal{Q}(\mathcal{D})$ while satisfying the autonomous constraint and security constraint in Definition 2.

For an exact query $\mathcal{D}$, the answer $\mathcal{Q}(\mathcal{D})$ should be equal to the (exact) result under an ideal case when all local datasets $\{\mathcal{D}_i\}$ have been integrated into one database $\mathcal{D}$. By contrast, for an approximate query $\mathcal{D}$, the answer $\mathcal{Q}(\mathcal{D})$ may be different from the (approximate) result under such a case. Instead, an approximation guarantee is often studied to control the result error for approximate queries.

Workflow of Processing Federated Queries. As shown in Figure 2, a service user first submits his query request $\mathcal{Q}$ to the central server (*a.k.a.*, coordinator or broker). Next, the server parses and rewrites the query $\mathcal{Q}$ to form a series of local queries and secure operations, which is generally aligned with our general framework in Section 2.2. Then, the server sends local queries to data owners, and asks them to perform local queries. When all of them have computed the local results, the server will coordinate these data owners to jointly perform secure operations by following the pre-designed protocols and obtain intermediate results or final results. The procedure ends after one or multiple rounds of local queries and secure operations, depending on the query type.

Here, we take the federated range counting query over a horizontal data federation as an example. As shown in Figure 1(a), data owners in a horizontal data federation hold different rows of the (virtual) table $\mathcal{D}$. According to the aforementioned workflow, a federated range counting query can be answered by performing a local range counting query over each local dataset $\mathcal{D}_i$ and a secure summation of these partial results across all data owners.

Intuitively, the performance of local queries and secure operations is critical to the performance of federated query processing. Thus, in the following, we introduce existing techniques for managing data federation from three categories, query rewrite (Section 3.2), local queries (Section 3.3), and secure operations (Section 3.4).

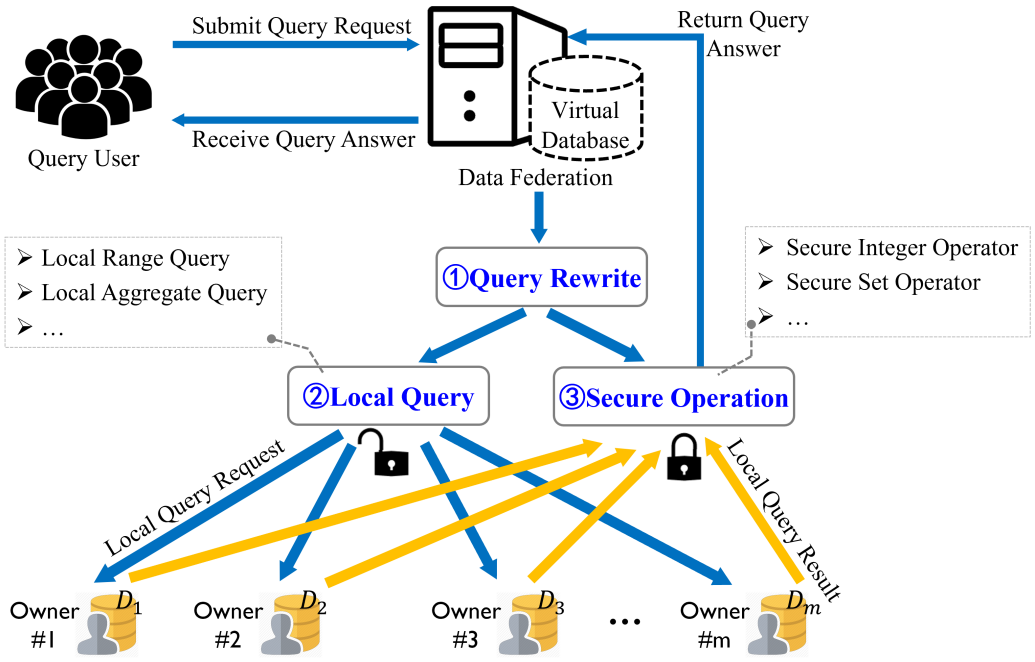


Figure 2: Workflow of processing federated queries over a data federation.

3.2 Query Rewrite

The *common idea* of query rewriters in federated queries is to use as many local queries and as few secure operations as possible [4]. This is because a secure operation across a large number of data owners is often more time-consuming than local queries that could be naturally sped up by modern database techniques. Based on this idea, existing query rewriter can be classified into two kinds, *query rewriter via annotating query plan tree* and *query rewriter via customized rule*.

Query Rewriter via Annotating Query Plan Tree. In this category, after receiving the query request, the coordinator will parse the federated query into a directed acyclic graph (DAG) by using well-known solutions (*e.g.*, Apache Calcite [11]). This DAG, which represents the query plan (when data is plaintext), is usually a tree of relational algebraic operations. After that, the coordinator will traverse the tree bottom-up (or sometimes up-down) to annotate the operations that require extra security protections (*i.e.*, secure operations).

Specifically, SMCQL [2], a system for processing federated queries, classified the levels of data access into three categories, public, protected, and private attributes. Here, public attributes are accessed by anyone (*e.g.*, the other data owners), and protected attributes could be accessed by the data owner, coordinator, and query user. By contrast, private value can be only accessed by the data owner himself. Accordingly, operators in the query plan tree can be categorized into three preservation kinds, *plaintext operators*, *secure operators*, and *prohibitive operators* (*i.e.*, three kinds of annotations). Then, the annotations can be determined by examining the data access levels of its output attributes and considering the most stringent level of its source attributes. For example, consider the federated top-k query over the attribute ID in the following:

```
SELECT ID FROM F ORDER BY ID LIMIT k;
```

(i) If ID is public, the query plan consists of two plaintext operators, *i.e.*, sort and limit k (by ID).

(ii) If ID is protected, the aforementioned two operators should be securely executed.

(iii) If ID is private, any query plan is prohibitive, since a private output attribute cannot be revealed to the service user.

To further optimize the query plan, Bater *et al.* [2] proposed a technique called *slicing* with the goal of minimizing the time cost of secure operations. They aimed to slice the secure operators into "smaller and more manageable units of computation" [2]. For instance, when ID is protected in the above example, one can slice the secure sort into *first* plaintext limit k over each data owner's local dataset and *then* secure sort over the remaining mk IDs, where m is the number of data owners. The query rewriters of other systems for federated queries, such as ShrinkWrap [12] and SAQE [13], followed a similar idea with that of SMCQL. The system Conclave [14] considered the scenario that a data owner may permit specific selectively-trusted parties (*e.g.*, a government regulator) to access his sensitive attributes and optimized the query rewriter under this assumption.

Query Rewriter via Customized Rule. Another way is to design customized query decomposition rules for specific query types. A typical example is the federated k-nearest-neighbor (kNN) query [4] in the spatial data federation. Although it is a spatial query, its query plan can be referred to that of federated top-k query, as aforementioned. Unfortunately, the efficiency has been shown to be low in large-scale datasets [4], since a secure sort is very time-consuming. Thus, Tong *et al.* [4] were motivated to design a novel query plan for federated kNN in their system Hu-Fu. The *basic idea* is to (1) determine the kth nearest distance to the query object by binary search and (2) retrieve the answer by a range query with the radius of the kth nearest distance. Accordingly, the query plan of the first step consists of local range counting queries (with the searching distance) and secure comparison (between the summation of all local counts and threshold k), and the query plan of the second step is the same as that of a federated range query. Another example is that Wang *et al.* [15] devised an efficient query plan for federated join-aggregate queries. Their basic idea is to decompose federated join-aggregate queries into semijoins and full joins.

3.3 Local Query

After getting the query plan, the coordinator often sends local queries to data owners prior to the secure operations. Thus, in the following, we introduce mainstream optimization techniques for processing local queries in a data federation from two aspects, *indexing* and *sampling*.

Optimization By Indexing. Indexing is usually used to improve the time efficiency of processing local queries. In particular, local queries in a data federation can be benefited from the recent development of indexing techniques. In recent years, the *learned index* is one of the most popular indexing techniques in existing literature. The *main idea* of a learned index is to view the data indexing problem as a machine learning problem that learns the mapping function between the search key and its corresponding location in the storage. Based on this idea, promising results of lookup queries and range queries over 1-dimensional data have been achieved by 1-dimensional learned indexes, such as RMI [16], PGM-index [17], ALEX [18], and LIPP [19]. Multi-dimensional learned indexes have also been devised to support kNN queries, such as ZM-Index [20], IF-Index [21], Lisa [22], and RSMI [23]. For more details, please refer to the tutorials [24] and experimental work [25].

Optimization By Sampling. Another way of optimizing local queries is to use sampling that may sacrifice result accuracy for efficiency. Such techniques in federated queries can be classified into two kinds, *sampling data owners* and *sampling data records*.

- **Sampling Data Owners.** The *main idea* of sampling data owners is to utilize the local results of the sampled data owners to estimate the local results of the others. For example, to process federated range aggregation queries over a spatial data federation, Shi *et al.* [26] used the result of a local range aggregation query over one data owner to derive an unbiased estimation of the others, where local datasets are identically and independently distributed (IID). To break the IID assumption and tackle the non-IID scenario, they proposed a grid index to decompose the underlying spatial area into small enough regions and achieve a

more fine-grained approximation by aggregating the estimation result in each region. They also proved that the estimated result can be closed enough to the exact result with high probability [26].

- **Sampling Data Records.** The *main idea* of sampling data records is to use the results of sampled data records to estimate the results of all data records within a local dataset. For example, Bater *et al.* [13] have applied three sampling strategies in their system for SAQE federated queries, *i.e.*, uniform sampling, stratified sampling, and distinct sampling, which are well-known sampling strategies in approximate query processing [27]. Shi *et al.* [26] adopted level sampling to achieve load balancing when performing local queries over unbalanced local datasets. Notice that, different from a traditional distributed database system, data partition or re-partition, which is a commonly-used technique for load balancing, is not allowed in a data federation, since each data owner would like to autonomously manage his own data.

In general, the strategy of sampling data owners is orthogonal to that of sampling data records. The technical challenge of jointly using both sampling strategies is how to make a proper trade-off between result accuracy and efficiency, especially under the non-IID scenario.

3.4 Secure Operation

After getting the results of local queries, secure operations are often invoked to jointly compute an intermediate result or a final result across all data owners and guarantee that no sensitive information (*e.g.*, private attributes) of one data owner is leaked to others during this collaborative computation. We introduce existing solutions to secure operations when processing federated queries from two aspects, *i.e.*, *secure multi-party computation (SMC) based solution* and *differential privacy (DP) based optimization*.

SMC based Solution. SMC has been studied for over three decades in academia [28]. The goal of SMC is to jointly compute some functions based on the private inputs of data owners in the data federation. Moreover, SMC also requires that (1) the result should be accurate and (2) no information can be leaked except for that derived from the output, which is coincident with the constraints of federated queries in Definition 3. Thus, SMC is a prevalent method to implement secure operations.

Specifically, the systems for federated queries, SMCQL [2] and ShrinkWrap [12], used a general-purpose programming framework of SMC (*a.k.a.*, SMC compilers), called ObliVM [29]. ObliVM mainly combines two well-known techniques in SMC, *i.e.*, Garbled Circuits (GC) [30] and Oblivious RAM (ORAM) [31]. GC can securely compute most of operations over two data owners, and ORAM is a safe memory abstraction to safely store the intermediate results of GC without leaking any information about the data access pattern. Other SMC compilers, such as Obliv-C [32] and Sharemind [33], are used in Conclave [14] to support two or three data owners. The other studies, such as [4] and [34], adopted customized SMC protocols to support specific operations, which could potentially improve the efficiency. Different from above systems, which assumed semi-honest adversaries, Senate [35] optimized a GC-based SMC protocol [36] to protect the security even with malicious data owners.

DP based Optimization. Differential privacy (DP) [37] is the state-of-the-art privacy protection technique to theoretically guarantee that one can hardly re-construct the database based on the query results by DP. In existing studies for federated queries, DP can be used to further improve the efficiency of secure operations by SMC. For example, ShrinkWrap [12] adopted DP to remove quite a few dummy tuples from the intermediate result in ORAM while still protecting the data access pattern. SAQE [12] used DP to hide the true cardinality of data records when performing oblivious sampling and perturb the query result. Hu-Fu [4] applied DP to speed up the time efficiency of secure comparison with the threshold k when processing federated kNN queries.

Beyond the above techniques, trusted hardware enclaves, such as Intel SGX [38], can be also used to implement secure operations in the system Opaque [39].

Table 1: Comparison of existing systems for federated queries

System	Data Type	Data Integration Mode	Attacker Model	Data Size	#(Data Owner)
SMCQL [2]	Relational	Horizontal	Semi-honest	$\leq$1K	$\leq$2
ShrinkWrap [12]	Relational	Horizontal	Semi-honest	$\leq$40K	$\leq$2
SAQE [13]	Relational	Horizontal	Semi-honest	$\leq$500K	$\leq$2
Conclave [14]	Relational	Horizontal/Vertical	Semi-honest	$\leq$1B	$\leq$3
Hu-Fu [4]	Spatial	Horizontal	Semi-honest	$\leq$1B	$\leq$10
Senate [35]	Relational	Horizontal	Malicious	$\leq$160K	$\leq$16
Opaque [39]	Relational	Vertical	Malicious	$\leq$1M	$\leq$5

3.5 Discussion

Table 1 summarizes the representative work on federated queries, and we have the following observations. First, most of the existing systems for federated queries concern more about semi-honest adversaries than malicious adversaries, although considering malicious adversaries are more challenging. Second, most of these studies focused on the horizontal data federation, while vertical data federation had less attention. Finally, the scalability of existing solutions is sometimes not large enough, especially when processing federated join queries.

4 Federated Learning

In this section, we introduce how to perform federated learning over a data federation. Specifically, we first present the problem statement and main workflow in Section 4.1. Then, we introduce the mainstream solutions to federated learning from two aspects, local training (Section 4.2) and secure aggregation (Section 4.3). Finally, we make discussions in Section 4.4.

4.1 Overview

Problem Statement. Federated learning (FL) was first proposed by Google in 2016 for language prediction tasks [6]. The *main idea* of federated learning is to train a global model in collaboration with different data owners while preserving the privacy of their local data. The definition of federated learning is as follows.

Definition 4 (Federated Learning): Given a data federation $\mathcal{F}$ of m data owners (*a.k.a.*, clients) $\{\mathcal{D}_i\}$ and a computing task of training a learning model $\mathcal{M}$ over the (virtual) database $\mathcal{D} = \cup_{i=1}^{m} \mathcal{D}_i$, a federated learning algorithm aims to collaboratively train the learning model $\mathcal{M}(\mathcal{D})$ on the data federation and make the accuracy of $\mathcal{M}(\mathcal{D})$ close to that of the model $\mathcal{M}$ that directly trains over the virtual database, while satisfying the autonomous constraint and security constraint in Definition 2.

Taxonomy. Federated learning algorithms can be categorized into three kinds, *horizontal federated learning*, *vertical federated learning*, and *federated transfer learning* [1] from the perspective of data integration mode across data owners.

- **Horizontal/Vertical Federated Learning.** Horizontal/vertical federated learning is named after the data integration mode (*a.k.a.*, data partition mode) in a data federation as shown in Figure 1. Notice that, for supervised learning tasks, all data owners' datasets in horizontal federated learning have labels, while only one of them in vertical federated learning has labels.

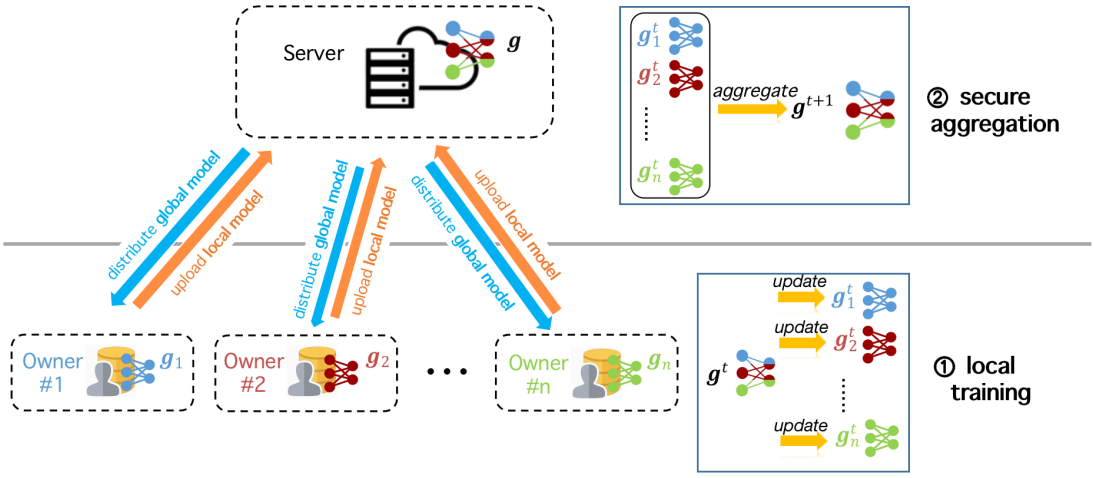


Figure 3: Workflow of federated learning over a data federation.

- **Federated Transfer Learning.** Federated transfer learning denotes the scenario where two data owners have different samples and feature spaces in the same time. In other words, federated transfer learning enables one data owner to make use of data from another data owner, although there is only little intersection between the feature spaces of their datasets.

Workflow of Federated Learning. According to the general framework of federated computing in Section 2.2, the workflow of federated learning algorithms is as follows. In the procedure of federated learning, the server first initializes a global model and distributes the global model to selected data owners. Each selected data owner then trains his local model through the local loss function and uploads his local model to the server. The server securely aggregates all the uploaded local models into the global model. Finally, the above steps are repeated until convergence.

A Case Study: FedAvg. One representative work in federated learning is FedAvg [6], which supports training several machine learning models, such as MLP, CNN and LSTM, in a data federation. Here, we take training a CNN model as an example to explain the aforementioned workflow of federated learning. After the server distributes the global model, data owners train their local CNN models with stochastic gradient descent (SGD). When the local training ends, data owners will upload the parameters of their local CNN models to the server. Then, the server aggregates all the received model parameters into a new global CNN model by weighted average. Now, Google has deployed such an algorithm in a text entry data prediction task on users' mobile devices.

Although the FedAvg algorithm [6] is a seminal work, there are still many opportunities for optimizations in federated learning. In the following, we review existing studies from two aspects, *i.e.*, local training (Section 4.2) and secure aggregation (Section 4.3).

4.2 Local Training

In this subsection, we introduce optimizing techniques for local training from two categories, *optimization through training* and *optimization through data*, as follows.

Optimization through Training. Optimization techniques through model training are widely used in recent years. It contributes to more stable convergence and better generalization of aggregated global models.

- **Regularized Loss.** The *main idea* of regularized loss methods in federated learning is to add a regularization term to local sub-problem and is widely used for limiting local model updates and stabilizing global model.

For example, FedProx [40] proposed a regularization term to take similarity between local model and global model into account. The regularization term keeps the differences between the local and global models within a certain range. FedDyn [41] introduces a dynamic regularization method, and ensures that the objective is dynamically updated and the local optima is close to the stationary point of the global empirical loss.

- **Extra Variable.** Some federated learning algorithms introduce extra variable to help improving model generalization. The *main difference* between regularized loss and extra variable is that the extra variable is usually updated during local updates. SCAFFOLD [42] set a control variable to correct the local update moving towards the true optimum. Several studies [43, 44] introduced dual variables, converting the local object at the client side (*i.e.*, the data owner side) into a dual problem. The optimization to the dual problem contributes to automatic adaptation to global data distributions.

Optimization through Data. Due to statistical heterogeneity caused by non-IID data across data owners, several data-based optimization techniques were proposed to alleviate the accuracy drop of a federated learning algorithm.

- **Data Augmentation.** Data augmentation, which enriches a dataset, is commonly used to fix the issue of non-IID data distribution on the client side to an IID one. FAug [45] trained a Generative Adversarial Network (GAN) at the server and distributed the GAN to clients, where GAN could generate data to the local dataset to achieve an IID data distribution. Fed-ZDAC [46] proposed a zero-shot data augmentation method, which synthesized data based on the model information (only) without sharing data with the server. Compared with FAug [45], Fed-ZDAC [46] achieved a higher privacy preservation level.

- **Data Source Selection.** During the training process of federated learning, some data owners may hold extremely skewed data, which may lead to slow and unstable convergence of the global model. The *main idea* of data source selection is to select proper data owners to participate the training, which alleviates the influence brought by skewed datasets. For instance, FDSS [47] observed the data source selection problem from the perspective of submodular optimization. By combining lazy evaluation and approximation of aggregated models with greedy selection, FDSS ensured a constant approximation ratio. Oort [48] modeled selecting data owners as a multi-armed bandit problem, and improved the time-to-accuracy performance of training procedure.

4.3 Secure Aggregation

While local training optimization contributes to better generalization and convergence, techniques of secure aggregation are important as well for satisfying the security constraint. Techniques of secure aggregation in federated learning can be categorized into two kinds, *secure multi-party computation (SMC) based solution* and *differential privacy (DP) based solution*.

SMC based Solution. Secure multi-party computation (SMC) enables several participants to collaboratively compute without revealing their data to each other. When the number of data owners is large, some SMC techniques, such as garbled circuit (GC) and oblivious transfer (OT), could be inefficient for secure aggregation in federated learning. Under this setting, efficient SMC based solutions are secret sharing (SS) and homomorphic encryption (HE).

- **Secret Sharing.** In a secret sharing scheme, a secret consists of multiple shares and can be re-constructed only when there is a sufficient number of shares. When secret sharing is used in a federated learning framework, the gradients of each data owner represent a secret and are re-constructed on the server side only when enough gradients are uploaded. Bonawitz *et al.* [49] presented a protocol to securely compute the sum of vectors, which has been used in federated learning. This protocol permits the server to securely

average updates uploaded by data owners without leaking these updates. By this way, secret sharing is commonly seen in federated learning algorithms [50, 51, 52] to protect the data privacy.

- **Homomorphic Encryption.** Homomorphic encryption guarantees security by conducting the calculation in ciphertext, and the result after decryption is same as that in plaintext calculation. For homomorphic encryption based secure aggregation in federated learning, data owners encrypt their local models and send them back to the server for later aggregation in ciphertext. Then, data owners decrypt the received global model and update local gradients. Several studies [53, 54, 55, 56] have adopted homomorphic encryption in the federated learning framework. BatchCrypt [57] introduced a batch encryption based technique to reduce the encryption and communication overhead caused by homomorphic encryption. It encodes a batch of gradients into a long integer data type to replace the original full precision encryption.

DP based Solution. Differential privacy (DP) protects privacy by adding perturbations to data. Compared with SMC, DP based solution is computationally more efficient. Thus, DP is widely used in federated learning to improve the efficiency. Existing DP based solution in federated learning can be mainly classified into two kinds, *central differential privacy (CDP)* and *local differential privacy (LDP)*.

- **Central Differential Privacy.** Central differential privacy provides a security guarantee by adding perturbation to the aggregated model on the server side. For example, Geyer *et al.* [58] proposed a central differential privacy based method. During the training stage, data owners update their model weights and upload parameters to the server. The server then aggregates model parameters with Gaussian noise. Based on a similar idea, Shi *et al.* [59] applied the central differential privacy to topic modeling, and NbAFL [60] added perturbation to local gradients in the meantime, and was shown to protect data owners from an untrusted server that planned to steal the gradients.

- **Local Differential Privacy.** By contrast, the local differential privacy based method adds noise to model parameters on the client side, such that model parameters can be prevented from being inferred by adversaries. For instance, Bao *et al.* [61] proposed a novel local differential privacy based federated learning framework, which injects noise from shifted symmetric Skellam distributions. It broadened the data type of model gradients, which contributed to a lower noise level required for differential privacy. Local differential privacy based solutions have also been adopted in several federated learning algorithms [60, 62, 63, 64] to achieve better protection for local model parameters and more scalable efficiency.

4.4 Discussion

In this subsection, we give a brief introduction on the comparison of representative work in Table 14. We can observe that more studies on federated learning were studied over the horizontal data federation, while less work studied vertical federated learning and federated transfer learning. Moreover, we can also observe that both SMC and DP strategies are used for satisfying the security constraint. By comparison, DP based solutions tend to achieve better scalability by supporting more data owners than SMC based solution. As shown in Figure 4, these techniques of federated learning have been integrated into algorithm framework by industry and academia, such as FATE [69], PySyft [70], and FederatedScope [71].

5 Future Direction

In the following, we identify the future directions of federated computing.

Exploring more federated queries and learning models. To facilitate more applications on more diversified data, it will be important to explore more federated queries and federated learning models.

Table 2: Comparison of representative work on federated learning.

Reference	Data Integration Mode	Local Training	Secure Aggregation	Learning Model	#(Data Owner)
NbAFL [60]	Horizontal	Regularized Loss	DP	MLP	≤50
SMM [61]	Horizontal	/	DP	MLP	≤240
FedADMM [44]	Horizontal	Extra Variable	/	CNN	≤100
FedAvg [6]	Horizontal	/	/	CNN, LSTM	≤600
FederatedScope [65]	Horizontal	/	SMC, DP	CNN, GNN	≤260
VF2Boost [66]	Vertical	/	SMC	GBDT	≤4
Pivot [67]	Vertical	/	SMC	GBDT	≤10
FTL [68]	Transfer*	Regularized Loss	SMC	AutoEncoder	≤2
FedHealth [55]	Transfer	Regularized Loss	SMC	CNN	≤25

* The term "Transfer" here represents federated transfer learning.

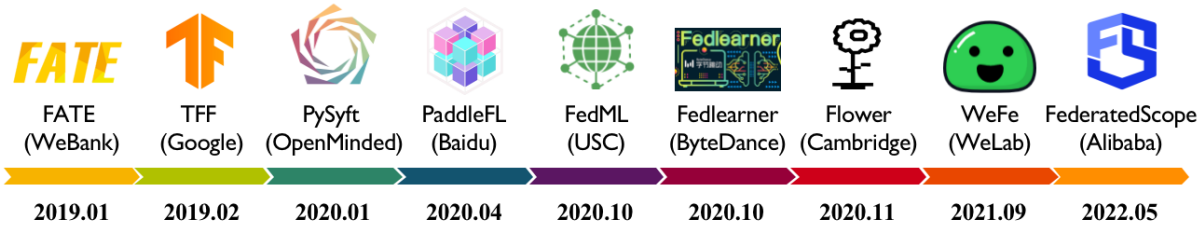


Figure 4: Representative frameworks for federated learning.

On one hand, as shown in Table 1, existing federated queries are mainly studied over relational data and spatial data. Other data types, such as trajectory data and graph data, are also important for data sharing. For example, Yuan *et al.* [5] have recently proposed the concept of graph data federation, and studied subgraph matching under this setting. Another application of graph data federation is a cross-platform ride-hailing [72], where each taxi company can be viewed as a data owner and their requests and passengers form a bipartite graph data federation. As a result, Wang *et al.* [72] studied how to obtain maximum weighted bipartite matching under a data federation. Graph data federation may have other applications (*e.g.*, social networks), and many other federated graph queries have not been studied yet, which leaves a great opportunity for future research.

On the other hand, although federated learning has been widely studied in the AI and data mining community, quite a few of them ignored the issue of data security. For instance, recent learning models, such as ViT [73], ResNets [74], and GraphSAGE [75], have been studied in the federated learning setting without protecting security and privacy rigorously. What is worse, recent surveys [76] have reviewed many FL attacks such as poisoning attacks. The attacks could probe and infer sensitive information from data owners' model parameters, leading to severe privacy leakage. Therefore, research on how to effectively defend these attacks for the aforementioned models in federated learning is still anticipated.

Marrying Federated Queries and Federated Learning. In the past five years, we have seen many promising results of marrying artificial intelligence (AI) and databases (DB), which is also known as DB4AI and AI4DB.

One typical example of AI4DB is the concept of learned index [16] that uses learning models to enhance, or even replace conventional indexes like B-Trees. Another example of DB4AI is the structure-aware learning system LMFAO [77] that decomposes the training procedure into batches of aggregate queries and further improves the efficiency by the optimization techniques of processing aggregate queries.

Motivated by these research trends, we envision that it is also possible that federated queries and federated learning could help each other. For instance, most of existing systems for federated queries have no support for a global index, which is often used to improve the query efficiency in a distributed DBMS. By contrast, in a data federation system for federated queries, a global index additionally needs to protect the sensitive information of all data owners. Since there are existing studies that have shown learned indexes can reduce the index size and running time, it might be possible that federated learning could be safely used to construct such a global index for federated queries.

Multi-Model Federated Computing. The variety is known as one of the fundamental challenges in managing big data. To fit a DBMS into diversified application settings, many data models have been proposed, such as relational, spatial, key/value, and graph. Intuitively, data-intensive applications, such as E-commerce [78] and transportation [79], may need to manipulate and analyze data with heterogeneous data models at the same time. Multi-model databases [78] have been proposed to tackle this problem. For example, PostgreSQL can now support several data models, including relational, key/value, JSON, XML, etc. However, most of these studies assume that all data have been collected and stored by only one data owner. By contrast, emerging applications have been deployed over a data federation.

On this basis, we propose a new concept called *multi-model big data federated computing ("multi-model federated computing" as short)* as the last line of future research. This computational paradigm aims to bridge the connections between data owners with diversified data models and provide joint queries and analytics while preserving data privacy/security. Under this setting, one fundamental challenge could be how to overcome the security heterogeneity [80] which inherits from data model variety. To the best of our knowledge, no existing work has built such a system, which leaves a valuable opportunity for future work.

6 Conclusion

Federated computing is a promising paradigm for data sharing, which enables secure querying and analysis across multiple autonomous data owners. This paper introduces the fundamental concepts and general framework of federated computing, along with discussions of the challenges and related studies on federated queries and federated learning. We also point out the future directions of federated computing and envision it as a practical solution to overcome the data isolation problem, fostering the prosperity of the information society.

Acknowledgements

This work is partially supported by National Science Foundation of China (NSFC) under Grant No. U21A20516 and 62076017, the Beihang University Basic Research Funding No. YWF-22-L-531, and WeBank Scholars Program. Yuxiang Zeng is the corresponding author.

References

[1] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," ACM Trans. Intell. Syst. Technol., vol. 10, no. 2, pp. 12:1–12:19, 2019.

[2] J. Bater, G. Elliott, C. Eggen, S. Goel, A. N. Kho, and J. Rogers, "SMCQL: secure query processing for private data networks," PVLDB, vol. 10, no. 6, pp. 673–684, 2017.

[3] A. Bharadwaj and G. Cormode, "An introduction to federated computation," in SIGMOD, 2022, pp. 2448–2451.

[4] Y. Tong, X. Pan, Y. Zeng, Y. Shi, C. Xue, Z. Zhou, X. Zhang, L. Chen, Y. Xu, K. Xu, and W. Lv, "Hu-Fu: Efficient and secure spatial queries over data federation," PVLDB, vol. 15, no. 6, pp. 1159–1172, 2022.

[5] Y. Yuan, D. Ma, Z. Wen, Z. Zhang, and G. Wang, "Subgraph matching over graph federation," PVLDB, vol. 15, no. 3, pp. 437–450, 2021.

[6] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in AISTATS, 2017, pp. 1273–1282.

[7] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. A. Bonawitz, Z. Charles, G. Cormode, R. Cummings, R. G. L. D'Oliveira, H. Eichner, S. E. Rouayheb, D. Evans, J. Gardner, Z. Garrett, A. Gascón, B. Ghazi, P. B. Gibbons, M. Gruteser, Z. Harchaoui, C. He, L. He, Z. Huo, B. Hutchinson, J. Hsu, M. Jaggi, T. Javidi, G. Joshi, M. Khodak, J. Konečný, A. Korolova, F. Koushanfar, S. Koyejo, T. Lepoint, Y. Liu, P. Mittal, M. Mohri, R. Nock, A. Özgür, R. Pagh, H. Qi, D. Ramage, R. Raskar, M. Raykova, D. Song, W. Song, S. U. Stich, Z. Sun, A. T. Suresh, F. Tramèr, P. Vepakomma, J. Wang, L. Xiong, Z. Xu, Q. Yang, F. X. Yu, H. Yu, and S. Zhao, "Advances and open problems in federated learning," Found. Trends Mach. Learn., vol. 14, no. 1-2, pp. 1–210, 2021.

[8] T. G. A. for Genomics and Health, "A federated ecosystem for sharing genomic, clinical data," Science, vol. 352, no. 6291, pp. 1278–1280, 2016.

[9] J. Vaidya and C. Clifton, "Privacy preserving association rule mining in vertically partitioned data," in SIGKDD, 2002, pp. 639–644.

[10] M. Kantarcioglu and C. Clifton, "Privacy-preserving distributed mining of association rules on horizontally partitioned data," IEEE Trans. Knowl. Data Eng., vol. 16, no. 9, pp. 1026–1037, 2004.

[11] E. Begoli, J. Camacho-Rodríguez, J. Hyde, M. J. Mior, and D. Lemire, "Apache calcite: A foundational framework for optimized query processing over heterogeneous data sources," in SIGMOD, 2018, pp. 221–230.

[12] J. Bater, X. He, W. Ehrich, A. Machanavajjhala, and J. Rogers, "Shrinkwrap: Efficient SQL query processing in differentially private data federations," PVLDB, vol. 12, no. 3, pp. 307–320, 2018.

[13] J. Bater, Y. Park, X. He, X. Wang, and J. Rogers, "SAQE: practical privacy-preserving approximate query processing for data federations," PVLDB, vol. 13, no. 11, pp. 2691–2705, 2020.

[14] N. Volgushev, M. Schwarzkopf, B. Getchell, M. Varia, A. Lapets, and A. Bestavros, "Conclave: secure multi-party computation on big data," in EuroSys, 2019, pp. 3:1–3:18.

[15] Y. Wang and K. Yi, "Secure Yannakakis: Join-aggregate queries over private data," in SIGMOD, 2021, pp. 1969–1981.

[16] T. Kraska, A. Beutel, E. H. Chi, J. Dean, and N. Polyzotis, "The case for learned index structures," in SIGMOD, 2018, pp. 489–504.

[17] P. Ferragina and G. Vinciguerra, "The PGM-index: a fully-dynamic compressed learned index with provable worst-case bounds," PVLDB, vol. 13, no. 8, pp. 1162–1175, 2020.

[18] J. Ding, U. F. Minhas, J. Yu, C. Wang, J. Do, Y. Li, H. Zhang, B. Chandramouli, J. Gehrke, D. Kossmann, D. B. Lomet, and T. Kraska, "ALEX: an updatable adaptive learned index," in SIGMOD, 2020, pp. 969–984.

[19] J. Wu, Y. Zhang, S. Chen, Y. Chen, J. Wang, and C. Xing, "Updatable learned index with precise positions," PVLDB, vol. 14, no. 8, pp. 1276–1288, 2021.

[20] H. Wang, X. Fu, J. Xu, and H. Lu, "Learned index for spatial queries," in MDM, 2019, pp. 569–574.

[21] A. Hadian, A. Kumar, and T. Heinis, "Hands-off model integration in spatial index structures," in AIDB@VLDB, 2020.

[22] P. Li, H. Lu, Q. Zheng, L. Yang, and G. Pan, "LISA: A learned index structure for spatial data," in SIGMOD, 2020, pp. 2119–2133.

[23] J. Qi, G. Liu, C. S. Jensen, and L. Kulik, "Effectively learning spatial indices," PVLDB, vol. 13, no. 11, pp. 2341–2354, 2020.

[24] S. Idreos and T. Kraska, "From auto-tuning one size fits all to self-designed and learned data-intensive systems," in SIGMOD, 2019, pp. 2054–2059.

[25] R. Marcus, A. Kipf, A. van Renen, M. Stoian, S. Misra, A. Kemper, T. Neumann, and T. Kraska, "Benchmarking learned indexes," PVLDB, vol. 14, no. 1, pp. 1–13, 2020.

[26] Y. Shi, Y. Tong, Y. Zeng, Z. Zhou, B. Ding, and L. Chen, "Efficient approximate range aggregation over large-scale spatial data federation," IEEE Trans. Knowl. Data Eng., vol. 35, no. 1, pp. 418–430, 2023.

[27] S. Chaudhuri, B. Ding, and S. Kandula, "Approximate query processing: No silver bullet," in SIGMOD, 2017, pp. 511–519.

[28] Y. Lindell, "Secure multiparty computation," Commun. ACM, vol. 64, no. 1, pp. 86–96, 2021.

[29] C. Liu, X. S. Wang, K. Nayak, Y. Huang, and E. Shi, "Oblivm: A programming framework for secure computation," in S&P, 2015, pp. 359–376.

[30] A. C. Yao, "Protocols for secure computations (extended abstract)," in FOCS, 1982, pp. 160–164.

[31] O. Goldreich and R. Ostrovsky, "Software protection and simulation on oblivious rams," J. ACM, vol. 43, no. 3, pp. 431–473, 1996.

[32] S. Zahur and D. Evans, "Obliv-C: A language for extensible data-oblivious computation," IACR Cryptology ePrint Archive, vol. 2015, p. 1153, 2015.

[33] D. Bogdanov, S. Laur, and J. Willemson, "Sharemind: A framework for fast privacy-preserving computations," in ESORICS, 2008, pp. 192–206.

[34] K. Zhang, Y. Tong, Y. Shi, Y. Zeng, Y. Xu, K. Xu, W. Lv, and Z. Zheng, "Approximate k-Nearest Neighbor Query over Spatial Data Federation,", in DASFAA, 2023, pp. 351–368.

[35] R. Poddar, S. Kalra, A. Yanai, R. Deng, R. A. Popa, and J. M. Hellerstein, "Senate: A maliciously-secure MPC platform for collaborative analytics," in USENIX Security, 2021, pp. 2129–2146.

[36] X. Wang, S. Ranellucci, and J. Katz, "Global-scale secure multiparty computation," in CCS, 2017, pp. 39–56.

[37] N. Li, M. Lyu, D. Su, and W. Yang, Differential Privacy: From Theory to Practice, ser. Synthesis Lectures on Information Security, Privacy, & Trust. Morgan & Claypool Publishers, 2016.

[38] W. Zheng, Y. Wu, X. Wu, C. Feng, Y. Sui, X. Luo, and Y. Zhou, "A survey of intel SGX and its applications," Frontiers Comput. Sci., vol. 15, no. 3, p. 153808, 2021.

[39] W. Zheng, A. Dave, J. G. Beekman, R. A. Popa, J. E. Gonzalez, and I. Stoica, "Opaque: An oblivious and encrypted distributed analytics platform," in NSDI, 2017, pp. 283–298.

[40] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," in MLSys, vol. 2, 2020, pp. 429–450.

[41] D. A. E. Acar, Y. Zhao, R. M. Navarro, M. Mattina, P. N. Whatmough, and V. Saligrama, "Federated learning based on dynamic regularization," in ICLR, 2021.

[42] S. P. Karimireddy, S. Kale, M. Mohri, S. J. Reddi, S. U. Stich, and A. T. Suresh, "SCAFFOLD: stochastic controlled averaging for federated learning," in ICML, vol. 119, 2020, pp. 5132–5143.

[43] Q. Tran-Dinh, N. H. Pham, D. T. Phan, and L. M. Nguyen, "FedDR - randomized Douglas-Rachford splitting algorithms for nonconvex federated composite optimization," in NeurIPS, 2021, pp. 30 326–30 338.

[44] Y. Gong, Y. Li, and N. M. Freris, "FedADMM: A robust federated deep learning framework with adaptivity to system heterogeneity," in ICDE, 2022, pp. 2575–2587.

[45] E. Jeong, S. Oh, H. Kim, J. Park, M. Bennis, and S. Kim, "Communication-efficient on-device machine learning: Federated distillation and augmentation under non-iid private data," CoRR, vol. abs/1811.11479, 2018.

[46] W. Hao, M. El-Khamy, J. Lee, J. Zhang, K. J. Liang, C. Chen, and L. Carin, "Towards fair federated learning with zero-shot data augmentation," in CVPR Workshops, 2021, pp. 3310–3319.

[47] R. Zhang, Y. Wang, Z. Zhou, Z. Ren, Y. Tong, and K. Xu, "Data source selection in federated learning: A submodular optimization approach," in DASFAA, vol. 13246, 2022, pp. 606–614.

[48] F. Lai, X. Zhu, H. V. Madhyastha, and M. Chowdhury, "Oort: Efficient federated learning via guided participant selection," in OSDI, 2021, pp. 19–35.

[49] K. A. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for privacy-preserving machine learning," in CCS, 2017, pp. 1175–1191.

[50] Y. Dong, X. Chen, L. Shen, and D. Wang, "Privacy-preserving distributed machine learning based on secret sharing," in ICICS, vol. 11999, 2019, pp. 684–702.

[51] G. Xu, H. Li, S. Liu, K. Yang, and X. Lin, "VerifyNet: Secure and verifiable federated learning," IEEE Trans. Inf. Forensics Secur., vol. 15, pp. 911–926, 2020.

[52] S. Sharma, C. Xing, Y. Liu, and Y. Kang, "Secure and efficient federated transfer learning," in IEEE BigData, 2019, pp. 2569–2576.

[53] L. T. Phong, Y. Aono, T. Hayashi, L. Wang, and S. Moriai, "Privacy-preserving deep learning via additively homomorphic encryption," IEEE Trans. Inf. Forensics Secur., vol. 13, no. 5, pp. 1333–1345, 2018.

[54] J. Zhang, B. Chen, S. Yu, and H. Deng, "PEFL: A privacy-enhanced federated learning scheme for big data analytics," in GLOBECOM, 2019, pp. 1–6.

[55] Y. Chen, X. Qin, J. Wang, C. Yu, and W. Gao, "FedHealth: A federated transfer learning framework for wearable healthcare," IEEE Intell. Syst., vol. 35, no. 4, pp. 83–93, 2020.

[56] Y. Zhang, Y. Shi, Z. Zhou, C. Xue, Y. Xu, K. Xu, and J. Du, "Efficient and Secure Skyline Queries over Vertical Data Federation," IEEE Trans. Knowl. Data Eng., pp. 1–12, 2023.

[57] C. Zhang, S. Li, J. Xia, W. Wang, F. Yan, and Y. Liu, "BatchCrypt: Efficient homomorphic encryption for cross-silo federated learning," in USENIX ATC, 2020, pp. 493–506.

[58] R. C. Geyer, T. Klein, and M. Nabi, "Differentially private federated learning: A client level perspective," CoRR, vol. abs/1712.07557, 2017.

[59] Y. Shi, Y. Tong, Z. Su, D. Jiang, Z. Zhou, and W. Zhang, "Federated Topic Discovery: A Semantic Consistent Approach," IEEE Intell. Syst., vol. 36, no.5, pp. 96–103, 2021.

[60] K. Wei, J. Li, M. Ding, C. Ma, H. H. Yang, F. Farokhi, S. Jin, T. Q. S. Quek, and H. V. Poor, "Federated learning with differential privacy: Algorithms and performance analysis," IEEE Trans. Inf. Forensics Secur., vol. 15, pp. 3454–3469, 2020.

[61] E. Bao, Y. Zhu, X. Xiao, Y. Yang, B. C. Ooi, B. H. M. Tan, and K. M. M. Aung, "Skellam mixture mechanism: a novel approach to federated learning with differential privacy," PVLDB, vol. 15, no. 11, pp. 2348–2360, 2022.

[62] R. Liu, Y. Cao, M. Yoshikawa, and H. Chen, "FedSel: Federated SGD under local differential privacy with top-k dimension selection," in DASFAA, vol. 12112, 2020, pp. 485–501.

[63] M. Seif, R. Tandon, and M. Li, "Wireless federated learning with local differential privacy," in ISIT, 2020, pp. 2604–2609.

[64] Y. Wang, Y. Tong, and D. Shi, "Federated Latent Dirichlet Allocation: A Local Differential Privacy Based Framework," in AAAI, 2020, pp. 6283–6290.

[65] Y. Xie, Z. Wang, D. Chen, D. Gao, L. Yao, W. Kuang, Y. Li, B. Ding, and J. Zhou, "Federatedscope: A flexible federated learning platform for heterogeneity," PVLDB, vol. 16, no. 5, p. 1059–1072, 2022.

[66] F. Fu, Y. Shao, L. Yu, J. Jiang, H. Xue, Y. Tao, and B. Cui, "VF^{2}Boost: Very fast vertical federated gradient boosting for cross-enterprise learning," in SIGMOD, 2021, pp. 563–576.

[67] Y. Wu, S. Cai, X. Xiao, G. Chen, and B. C. Ooi, "Privacy preserving vertical federated learning for tree-based models," PVLDB, vol. 13, no. 11, pp. 2090–2103, 2020.

[68] Y. Liu, Y. Kang, C. Xing, T. Chen, and Q. Yang, "A secure federated transfer learning framework," IEEE Intell. Syst., vol. 35, no. 4, pp. 70–82, 2020.

[69] "FATE," https://github.com/FederatedAI/FATE, last accessed 28 Feb 2023.

[70] "PySyft," https://github.com/OpenMined/PySyft, last accessed 28 Feb 2023.

[71] "FederatedScope," https://github.com/alibaba/FederatedScope, last accessed 28 Feb 2023.

[72] Y. Wang, Y. Tong, Z. Zhou, Z. Ren, Y. Xu, G. Wu, and W. Lv, "Fed-LTD: Towards cross-platform ride hailing via federated learning to dispatch," in SIGKDD, 2022, pp. 4079–4089.

[73] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," in ICLR, 2021.

[74] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in CVPR, 2016, pp. 770–778.

[75] W. L. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in NeurIPS, 2017, pp. 1024–1034.

[76] X. Yin, Y. Zhu, and J. Hu, "A comprehensive survey of privacy-preserving federated learning: A taxonomy, review, and future directions," ACM Comput. Surv., vol. 54, no. 6, pp. 131:1–131:36, 2022.

[77] D. Olteanu, "The relational data borg is learning," PVLDB, vol. 13, no. 12, pp. 3502–3515, 2020.

[78] J. Lu and I. Holubová, "Multi-model databases: A new journey to handle the variety of data," ACM Comput. Surv., vol. 52, no. 3, pp. 55:1–55:38, 2019.

[79] J. Lu, I. Holubová, and B. Cautis, "Multi-model databases and tightly integrated polystores: Current practices, comparisons, and open challenges," in CIKM, 2018, pp. 2301–2302.

[80] Y. Cao, W. Fan, Y. Wang, and K. Yi, "Querying shared data with security heterogeneity," in SIGMOD, 2020, pp. 575–585.

Federated Learning without Full Labels: A Survey

Yilun Jin[†] Yang Liu[‡] Kai Chen[†] Qiang Yang[†]
[†] Department of CSE, HKUST, Hong Kong, China
yilun.jin@connect.ust.hk, {qyang,kaichen}@cse.ust.hk
[‡] Institute for AI Industry Research, Tsinghua University, Beijing, China
liuy03@air.tsinghua.edu.cn

Abstract

Data privacy has become an increasingly important concern in real-world big data applications such as machine learning. To address the problem, federated learning (FL) has been a promising solution to building effective machine learning models from decentralized and private data. Existing federated learning algorithms mainly tackle the supervised learning problem, where data are assumed to be fully labeled. However, in practice, fully labeled data is often hard to obtain, as the participants may not have sufficient domain expertise, or they lack the motivation and tools to label data. Therefore, the problem of federated learning without full labels is important in real-world FL applications. In this paper, we discuss how the problem can be solved with machine learning techniques that leverage unlabeled data. We present a survey of methods that combine FL with semi-supervised learning, self-supervised learning, and transfer learning methods. We also summarize the datasets used to evaluate FL methods without full labels. Finally, we highlight future directions in the context of FL without full labels.

1 Introduction

Deep learning (DL) algorithms have achieved great success in the past decade. Powered by large-scale data such as ImageNet [1], ActivityNet [2], BookCorpus [3], and WikiText [4], deep learning models have been successfully applied to image classification [5], object detection [6], and natural language understanding [7]. However, the success of DL relies on large-scale, high-quality data, which is not always available in practice for two reasons. On one hand, collecting and labeling data is costly, making it difficult for a single organization to accumulate and store large-scale data. On the other hand, it is also infeasible to share data across organizations to build large-scale datasets, as doing so leads to potential leakage of data privacy. In recent years, a series of laws and regulations have been enacted, such as the General Data Protection Regulation (GDPR) [8] and the California Consumer Privacy Act (CCPA) [9], imposing constraints on data sharing. Therefore, how to jointly leverage the knowledge encoded in decentralized data while protecting data privacy becomes a critical problem.

Federated Learning (FL) [10, 11] is a promising solution to the problem and has received great attention from both the industry and the research community. The key idea of FL is that *participants* (also known as *clients* or *parties*) exchange intermediate results, such as model parameters and gradients, instead of raw data, to jointly train machine learning models. As the raw data never leave their owners during model training, FL becomes an

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

attractive privacy-preserving solution to the problem of decentralized machine learning. Up to now, a plethora of FL techniques has been proposed, focusing primarily on addressing the issues of data heterogeneity [13, 15], system heterogeneity [14, 17], data privacy and security [16, 18], and communication efficiency [12, 19].

Despite the significant research efforts, there is still one important yet under-explored topic in FL, which is how to effectively leverage unlabeled data to learn better federated models. In existing efforts of FL [12, 13, 14], it is assumed that all data held by all participants are fully labeled, and that a supervised learning problem is to be solved. However, the assumption may not hold in practice for two reasons. First, participants may not be sufficiently motivated to label their data. For example, suppose a sentiment classification model is to be trained with FL, smartphone users would be unwilling to spend time and effort to label all sentences typed in the phone. Second, participants may not have sufficient expertise to label their data. For example, wearable devices record various data (e.g. heart rate, breath rate, etc.) about the user's physical conditions, labeling which would require domain expertise in medical science and cannot be done by ordinary users. Based on the above observations, we argue that unlabeled data widely exist in real-world FL applications, and that the problem of *Federated Learning without Full Labels* is an important problem to study.

There are generally three learning paradigms in centralized machine learning (ML) that tackle the problem of learning without full labels, *semi-supervised learning* [20, 21], *self-supervised learning* [24, 23], and *transfer learning* [22], all of which have drawn much attention from researchers. Among them, semi-supervised learning aims to leverage unlabeled data to assist the limited labeled data [25, 26, 27]. Self-supervised learning aims to learn indicative feature representations from unlabeled data, which are then used to assist downstream supervised learning tasks [28, 29, 30]. Transfer learning aims to use sufficient data from a *source* domain to assist learning in a *target* domain with insufficient data [31, 32, 33], where the target domain commonly contains unlabeled data. However, despite the large number of existing works in these areas, it is not straightforward to apply them in FL due to the following challenges.

- **Isolation of labeled and unlabeled data.** In traditional semi-supervised learning and transfer learning, the server has access to both labeled and unlabeled data. However, in FL without full labels, it is common for a participant to have unlabeled data only. For example, a medical institute may not have the expertise to diagnose a complex illness, leaving all its data unlabeled. Moreover, it is not allowed in FL to exchange labeled data to solve the problem. The isolation of labeled and unlabeled data may compromise the overall performance. As observed in [34, 35], training with only unlabeled data leads to forgetting the knowledge learned from labeled data, which negatively impacts the overall performance. Therefore, it is important to bridge the knowledge between labeled and unlabeled data, without data exchange.

- **Privacy of labeled data.** In the problem of FL without full labels, the number of labeled data is often limited. Therefore, participants have to repetitively access and exchange information about them to exploit the knowledge in the labels. This leads to risks of privacy leakage of the labeled data. For example, semi-honest participants can learn to reconstruct the labeled data via gradient inversion attacks [36].

- **Data heterogeneity.** Data heterogeneity, i.e. the local data held by different participants have different data distributions, is an important property in FL that causes accuracy degradation [13, 14]. Similarly, data heterogeneity also poses challenges in the problem of FL without full labels. For example, as the number of labeled data is limited, local models tend to overfit the local data more easily, which causes a greater amount of weight divergence [37] and performance degradation.

- **Balancing performance and efficiency.** The large-scale unlabeled data in the problem creates a tradeoff between performance and efficiency. Specifically, while large-scale unlabeled data is available for training, their impacts on the model performance may be marginal, and the overall efficiency can be improved by sampling a fraction of unlabeled data without compromising model performance.

In this paper, we present a survey of the problem of FL without full labels and its existing solutions. The rest of the paper is organized as follows. Section 2 presents necessary backgrounds about FL as well as machine learning paradigms without full labels, including semi-supervised learning, self-supervised learning, and transfer learning. Sections 3, 4, 5 then review methods on federated semi-supervised learning, federated self-supervised learning, and federated transfer learning, respectively. Section 6 summarizes the datasets used for evaluating FL methods without full labels. Section 2 analyzes the similarities and differences between our work and related surveys. Finally, Section 8 presents an outlook on potential directions in the context of FL without full labels.

2 Preliminaries

In this section, we formally introduce backgrounds about FL, as well as the machine learning paradigms leveraging unlabeled data, semi-supervised learning, self-supervised learning, and transfer learning.

2.1 Federated Learning (FL)

Federated Learning aims to virtually unify decentralized data held by different participants to train machine learning models while protecting data privacy. Depending on how the data is split across participants, FL can be divided into horizontal federated learning (HFL) and vertical federated learning (VFL) [10]. In HFL, participants own data with the same feature space (e.g. participants own image data from different users), while in VFL, participants own data with the same user space but different feature spaces (e.g. a financial institute owns transaction records of a user, while an e-commerce corporation owns purchase records). In this paper, following the majority of existing research efforts, we primarily focus on HFL[1], i.e. all participants share the same feature space. Formally, we consider an FL scenario with C participants, denoted as $1, \ldots, C$. Each participant i owns a dataset $\mathcal{D}_i = \{\mathbf{X}_{ij}, y_{ij}\}_{j=1}^{N_i}$, where $N_i = |\mathcal{D}_i|$ is the number of data held by participant i, and $\mathbf{X}_{ij}, y_{ij}$ denote the features and the label of the j-th sample from client i, respectively. We use $p_i(\mathbf{X}, y)$, $p_i(\mathbf{X})$, $p_i(y|\mathbf{X})$ to denote the joint distribution, marginal distribution, and conditional distribution of client i, respectively. Denoting the model parameters as $\theta \in \mathbb{R}^d$, the overall optimization objective of FL is as follows,

$$\min_{\theta} f_{fl}(\theta) = \frac{1}{C}\sum_{i=1}^{C} f_{fl,i}(\theta), \text{where } f_{fl,i}(\theta) = \frac{1}{N_i}\sum_{j=1}^{N_i} l\left(\mathbf{X}_{ij}, y_{ij}; \theta\right), \text{s.t. } M_p(\theta) < \varepsilon_p, \tag{1}$$

where $f_{fl,i}(\theta)$ is the local optimization objective of participant i, and $l(\mathbf{X}, y; \theta)$ is a loss function, such as the cross-entropy loss for classification problems. In addition, $M_p(\theta)$ denotes a metric measuring the privacy leakage of θ (e.g. the budget in differential privacy (DP) [72]), and ε_p is a privacy constraint.

The training process of FL generally involves multiple communication rounds, each of which contains two steps, local training, and server aggregation.

- In the local training stage, a subset of all participants is selected. They are given the latest global model and will train the model with their local data for several epochs.
- In the server aggregation stage, participants upload their updated parameters to the server. The server aggregates received parameters via weighted averaging to obtain the global model for the next round.

Depending on the properties of participants, FL can be categorized into *cross-device* FL and *cross-silo* FL [11]. Participants of cross-device FL are commonly smart devices (e.g. phones, sensors, wearables) connected with wireless networks, while participants of cross-silo FL are commonly large organizations with connected datacenters, implying the following differences:

[1] Unless otherwise specified, we will use FL to refer to HFL throughout this paper.

- *Computation/communication capability*. Participants in cross-device FL commonly have limited computation (e.g. small memory, limited power supply) and communication capability (e.g. wireless network).
- *Stability*. Participants in cross-device FL are not stable and may drop out due to network breakdown.
- *Participant states*. In general, participants in cross-device FL cannot carry state vectors, in that they may only participate in one round of FL, and then drop out indefinitely.

2.2 Machine Learning with Unlabeled Data

2.2.1 Semi-supervised Learning

In semi-supervised learning, there are two datasets, a labeled dataset $\mathcal{L} = \{\mathbf{X}_j, y_j\}_{j=1}^{|\mathcal{L}|}$, and an unlabeled dataset $\mathcal{U} = \{\mathbf{X}_k\}_{k=1}^{|\mathcal{U}|}$, $|\mathcal{U}| \ll |\mathcal{L}|$. In addition, the marginal distributions of $\mathcal{L}, \mathcal{U}$ are the same, i.e. $p_{\mathcal{L}}(\mathbf{X}) = p_{\mathcal{U}}(\mathbf{X})$. The goal of semi-supervised learning, involving both labeled and unlabeled data, is as follows,

$$\min_{\theta} f_{semi}(\theta) = \frac{1}{|\mathcal{L}|} \sum_{j=1}^{|\mathcal{L}|} l_s(\mathbf{X}_j, y_j; \theta) + \frac{1}{|\mathcal{U}|} \sum_{k=1}^{|\mathcal{U}|} l_u(\mathbf{X}_k; \theta), \tag{2}$$

where l_s, l_u denotes the loss for labeled (supervised) and unlabeled data, respectively.

We then introduce some widely adopted techniques in semi-supervised learning.

Pseudo-Labeling [79]. Pseudo-labeling is a simple but effective trick for semi-supervised learning. Specifically, for each unlabeled data sample, its pseudo-label is taken as the class with the highest predicted probability,

$$\hat{y}_k = \arg\max_c g_\theta(\mathbf{X}_k)_c, \tag{3}$$

where g_θ is the model with parameter θ, and $g_\theta(\mathbf{X}_k)_c$ denotes the predicted probability of class c for $\mathbf{X}_k$. There is often a confidence threshold τ, such that pseudo-labels are only taken on confident samples with $\hat{y}_k > \tau$. After that, the pseudo-labels are used to supervise learning on unlabeled data, i.e.

$$l_u(\mathbf{X}_k; \theta) = l_s(\mathbf{X}_k, \hat{y}_k; \theta). \tag{4}$$

Teacher-student Models [25]. Teacher-student models in semi-supervised learning leverage two networks, a teacher model θ_{tea} and a student model θ_{stu}. On one hand, the student model is trained to be consistent with the teacher model to enhance its robustness

$$l_u(\mathbf{X}_k; \theta) = d\left(g_{\theta_{stu}}(\mathbf{X}_k), g_{\theta_{tea}}(\mathbf{X}_k)\right), \tag{5}$$

where $d(\cdot, \cdot)$ is a distance metric. On the other hand, the teacher model is updated with moving averaging (parameterized by α) over the student model after each iteration

$$\theta_{tea} = (1 - \alpha)\theta_{tea} + \alpha\theta_{stu}. \tag{6}$$

2.2.2 Self-supervised Learning

Self-supervised learning aims to learn good feature representations from unlabeled data to facilitate downstream machine learning tasks. There are in general two ways to perform self-supervised learning, generative learning, and contrastive learning [24]. Generative learning trains the model to reconstruct the original data $\mathbf{X}$ from masked data to learn the internal semantics within $\mathbf{X}$, while contrastive learning trains the model to distinguish between

Machine Learning Paradigm	Assumptions		Application	Limitations
	Train-test i.i.d.	Labeled Data		
Supervised Learning	✓	✓	Sufficient labeled data	Labeled data is hard to obtain.
Semi-supervised Learning	✓	Insufficient	A few labeled data + large-scale unlabeled data	Labeled and unlabeled data should have the same distribution.
Self-supervised Learning	✓	×	Large-scale unlabeled data	Cannot directly perform supervised tasks.
Transfer Learning	×	From another domain	Unlabeled data + labeled data *from another domain*	Hard to select a helpful source domain. Potential negative transfer.

Table 3: A comparison between supervised learning, semi-supervised learning, self-supervised learning, and transfer learning.

'positive' and 'negative' samples. In this survey, we primarily focus on contrastive learning, whose objective is given as follows.

$$\min_{\theta} f_{ctr}(\theta) = \sum_{\mathbf{X}\in\mathcal{U}} \left[d\left(g_\theta(\mathbf{X}), g_\theta(\mathbf{X}_+)\right) - \lambda \cdot d\left(g_\theta(\mathbf{X}), g_\theta(\mathbf{X}_-)\right)\right], \tag{7}$$

where $\mathcal{U}$ is the unlabeled dataset, g_θ is a neural network parameterized by θ, $\mathbf{X}_+, \mathbf{X}_-$ are positive and negative samples sampled for data $\mathbf{X}$, λ is the weight for negative samples, and $d(\cdot,\cdot)$ is a distance metric. By minimizing f_{ctr}, the model g_θ learns to minimize the distance between positive samples in the feature space, while maximizing the distance between negative ones. Some representative contrastive learning methods include SimCLR [45], MoCo [46], BYOL [48], and SimSiam [47]. We briefly explain their similarities and differences.

Similarities. All four methods employ a Siamese structure – two networks with the same architecture. One of them is called the online network θ_o and the other is called the target network θ_{tar}. The main difference is that the online network is directly updated via gradient descent, while the target network is generally not.

Differences. The differences between existing self-supervised learning methods are generally three-fold.

1. **Architecture**. In SimCLR and MoCo, the online and the target networks have the same architecture. On the contrary, for SimSiam and BYOL, the online network contains an additional predictor, i.e. $\theta_o = (\theta_o^f, \theta_o^p)$. The predictor aims to transform features between different views, enabling additional diversity.

2. **Target Network Parameter**. For SimCLR and SimSiam, the target network shares the same parameters as the online network $\theta_o = \theta_{tar}$, while for BYOL and MoCo, the target network is updated with an exponential moving average similar to Eqn. 6.

3. **Negative Samples**. On one hand, SimCLR and MoCo require negative samples $\mathbf{X}_-$. MoCo generates negative samples from previous batches, while SimCLR takes all other samples in the same batch as negative samples. On the other hand, SimSiam and BYOL do not require negative samples (i.e. $\lambda = 0$).

2.2.3 Transfer Learning

Both semi-supervised learning and self-supervised learning assume that the training and test data are independent and identically distributed (i.i.d.), regardless of whether labels are present. However, transfer learning [22] does not require the assumption. Specifically, transfer learning deals with multiple data distributions (also called *domains*) $p_i(\mathbf{X}, y), i = 1, 2, \dots T$, where the model is trained on one, and tested on another. Without loss of generality, we assume that $T = 2$. We denote $\mathcal{L}_1 = \{\mathbf{X}_{1i}, y_{1i}\}_{i=1}^{|\mathcal{L}_1|} \sim p_1(\mathbf{X}, y)$ as the *source* dataset, and $\mathcal{U}_2 = \{\mathbf{X}_{2j}\}_{j=1}^{|\mathcal{U}_2|} \sim p_2(\mathbf{X})$ as the *target* dataset. The overall goal is to minimize the error on the target dataset. However, as there are no labeled target data, we resort to the abundant source data to learn a model that generalizes

well to the target dataset. A commonly studied optimization objective is as follows,

$$\min_{\theta_f,\theta_c} \underbrace{\sum_{i=1}^{|\mathcal{L}_1|} l_s(\mathbf{X}_{1i}, y_{1i}; \theta_f, \theta_c)}_{f_{cls}(\mathcal{L}_1;\theta_f,\theta_c)} + \lambda \cdot \underbrace{d\left(g_{\theta_f}\left(\mathcal{L}_1\right), g_{\theta_f}\left(\mathcal{U}_2\right)\right)}_{f_{dom}(\mathcal{L}_1,\mathcal{U}_2;\theta_f)}, \tag{8}$$

where θ_f, θ_c, are parameters of the feature extractor and the classifier, respectively, $d(\cdot,\cdot)$ is a distance metric, $g_{\theta_f}(\mathcal{L}_1) = \{g_{\theta_f}(\mathbf{X}_{1i})\}_{i=1}^{|\mathcal{L}_1|}$ denotes the set of source features extracted by θ_f, and f_{cls}, f_{dom} denote the classifier loss on the source domain and the domain distance between domains, respectively. Intuitively, Eqn. 8 aims to minimize the classification error on the source domain, while minimizing the distance between source domain features and target domain features. In this way, the feature extractor θ_f is considered to extract domain-invariant features, and the classifier can be reused in the target domain. Commonly used distance metrics $d(\cdot,\cdot)$ include L_2 distance, maximum mean discrepancy (MMD) [32] and adversarial domain discriminator [33].

In addition, if an additional labeled target dataset $\mathcal{L}_2$ is available, θ_f, θ_c can be further fine-tuned with $\mathcal{L}_2$.

Transfer learning can generally be categorized into *homogeneous transfer learning* and *heterogeneous transfer learning* [22]. Homogeneous transfer learning assumes that domains share the same feature and label space, while heterogeneous transfer learning does not make such an assumption. For example, consider a movie recommender system that would like to borrow relevant knowledge from a book recommender system. If both systems rely on text reviews and ratings for recommendation, then a homogeneous transfer learning is to be solved, with the shared feature space being texts, and the shared label space being the ratings. However, if the movie recommender wants to leverage additional video clips, then the problem becomes a heterogeneous transfer learning problem, as the book recommender does not have video features. Heterogeneous transfer learning generally requires explicit cross-domain links to better bridge heterogeneous features and labels. For example, a novel and its related movie products should have similar feature representations.

2.2.4 Summary and Discussion

We summarize the three learning paradigms involving unlabeled data in Table 3. As shown, supervised learning has two key assumptions, the i.i.d. property between training and test data, and sufficient labeled data. Therefore, supervised learning is not applicable when either the labeled data is insufficient, or the training and test data come from different distributions. To address the drawback, semi-supervised learning, self-supervised learning, and transfer learning are proposed to relax the two key assumptions.

- Semi-supervised learning relaxes the assumption of sufficient labeled data. With limited labeled data, semi-supervised learning aims to exploit large-scale unlabeled data that have the same distribution as labeled data with techniques such as pseudo-labeling or teacher-student models. The main limitation of semi-supervised learning is the difficulty to obtain i.i.d. unlabeled data. For example, for the task of medical imaging, the images taken from multiple hospitals may follow different distributions due to device differences, demographic shifts, etc.

- Self-supervised learning further relaxes the assumption of labeled data. It aims to learn meaningful feature representations from the internal structures of unlabeled data, such as patches, rotations, and coloring in images. The main limitation of self-supervised learning is that, although it does not require labels to learn feature representations, they cannot be directly used to perform supervised tasks (e.g. classification).

- Transfer learning further relaxes the assumption of i.i.d. train and test data. Given unlabeled data in a domain, it aims to learn from a different but related domain with sufficient labeled data, and to transfer helpful knowledge to the unlabeled data. The main limitation of transfer learning is that it commonly

requires trial-and-errors to select an adequate source domain. When inadequate source domains are chosen, negative transfer [83] may happen which compromises model accuracy.

3 Federated Semi-supervised Learning

In this section, we present an overview of federated semi-supervised learning, whose main goal is to jointly use both labeled and unlabeled data owned by participants to improve FL. Before introducing detailed techniques, we first categorize federated semi-supervised learning into two settings following [35]:

- **Label-at-client**, where the labeled data are located at the clients, while the server only has access to unlabeled data. For example, when a company would like to train an FL model for object detection using images taken from smartphones, the company has no access to the local data of users, and labeling can only be done by users. However, users are generally unwilling to label every picture taken from their smartphones, creating a label-at-client setting for federated semi-supervised learning. Formally, the objective function of this setting is as follows,

$$\min_{\theta} \frac{1}{C} \sum_{i=1}^{C} f_{semi,i}(\theta), \text{s.t. } M_p(\theta) < \varepsilon_p \tag{9}$$

  where $f_{semi,i}(\theta)$ denotes the semi supervised learning loss (Eqn. 2) evaluated on the dataset of participant i, and M_p, ε_p follow Eqn. 1.

- **Label-at-server**, where the labeled data are located at the server, while clients have only unlabeled data. For example, consider a company of wearable devices that would like to train a health condition monitoring model with FL. In this case, users generally do not have the expertise to label data related to health conditions, leaving the data at clients unlabeled. The objective can be similarly formulated as

$$\min_{\theta} \frac{1}{|\mathcal{L}|} \sum_{j=1}^{|\mathcal{L}|} l_s(\mathbf{X}_j, y_j; \theta) + \frac{1}{C} \sum_{i=1}^{C} \left(\frac{1}{|\mathcal{U}_i|} \sum_{k=1}^{|\mathcal{U}_i|} l_u(\mathbf{X}_{ik}; \theta) \right), \text{s.t. } M_p(\theta) < \varepsilon_p. \tag{10}$$

Methods for each federated semi-supervised learning setting are discussed in the following sections. We also summarize existing methods in Table 4.

3.1 The Label-at-client Setting

The label-at-client setting of federated semi-supervised learning is similar to conventional FL (Eqn. 1), in that clients can train local models with their labeled data, and the updated parameters are aggregated by the server. Therefore, the label-at-client setting inherits the challenges of data heterogeneity, data privacy, and efficiency tradeoff from conventional FL. In addition, some clients may not have labeled data to train their local models, causing the label isolation problem. We introduce how existing works address these problems in this section.

RSCFed [38] primarily focuses on the label isolation problem and the data heterogeneity problem in federated semi-supervised learning. For local training, the teacher-student model (introduced in Section 2.2.1) is adopted for training on unlabeled data. To further address the data heterogeneity problem, RSCFed proposes a sub-consensus sampling method and a distance-weighted aggregation method. In each round, several sub-consensus models are aggregated by independently sampling multiple subsets of all participants, such that each sub-consensus model is expected to contain participants with labeled data. Moreover, the local models are weighted according to their distance to sub-consensus models, such that deviating models receive low weights and their impacts are minimized.

Setting	Method	Label Isolation	Data Privacy	Data Heterogeneity	Efficiency Tradeoff
Label-at-client	RSCFed [38]	Teacher-student model	×	Sub-consensus models & distance-weighted aggregation	×
	FedSSL [39]	Pseudo-labeling	Differential privacy (DP)	Global generative model	×
	FedMatch [35]	Pseudo-labeling	×	Inter-client consistency	Disjoint & sparse learning
	FedPU [41]	Negative labels from other clients	×	×	×
	AdaFedSemi [40]	Pseudo-labeling	×	×	Tuning confidence threshold and participation rate.
	DS-FL [42]	Ensemble pseudo-labeling	×	Entropy reduction averaging	Transmit logits, not parameters
Label-at-server	SemiFL [34]	Alternate training & Pseudo-labeling	×	×	×
	FedMatch [35]	Pseudo-labeling & Disjoint learning	×	Inter-client consistency loss	Disjoint & sparse learning

Table 4: Summary of techniques for federated semi-supervised learning. × indicates that the proposed method does not focus on this issue.

FedSSL [39] tackles the label isolation problem, the data privacy problem, and the data heterogeneity problem. To facilitate local training of unlabeled clients, FedSSL leverages the technique of pseudo-labeling. Further, to tackle the data heterogeneity problem, FedSSL learns a global generative model to generate data from a unified feature space, such that the data heterogeneity is mitigated by the generated data. Finally, to prevent privacy leakage caused by the generative model, FedSSL leverages differential privacy (DP) to limit the information leakage of the training data in the generative model.

FedMatch [35] proposes an inter-client consistency loss to address the data heterogeneity problem. Specifically, top-k nearest clients are sampled for each client, and on each data sample, the output of the local model is regularized with those of the top-k client models to ensure consistency. In addition, FedMatch proposes disjoint learning that splits the parameters for labeled and unlabeled data, and the parameters for unlabeled data are sparse. Upon updates, clients with only unlabeled data upload sparse tensors, reducing the communication cost.

FedPU [41] studies a more challenging setting within semi-supervised learning, positive and unlabeled learning, in which each client has only labels in a subset of classes. In this setting, a client has only information about a part of all classes, leading to a severe label isolation problem. To tackle the problem, FedPU derives a novel objective function, such that the task of learning the negative classes of a client is relegated to other clients who have labeled data in the negative class. In this way, each client is only responsible for learning the positive classes and can do local training by itself. Empirically, the proposed FedPU outperforms FedMatch [35] in the positive-and-unlabeled learning setting.

AdaFedSemi [40] proposes a system to achieve the tradeoff between efficiency and model accuracy in federated semi-supervised learning with server-side unlabeled data. For every round, the model is trained with labeled data at clients and aggregated at the server. The server-side unlabeled data are incorporated into the training process via pseudo-labeling. AdaFedSemi [40] identifies two key parameters to balance the tradeoff between efficiency and performance, the client participation rate P, and the confidence threshold of pseudo-labels τ. A lower P reduces both the communication cost and the model accuracy, while a high τ reduces the server-side computation cost while also limiting the usage of unlabeled data. Therefore, AdaFedSemi designs a tuning method based on multi-armed bandits (MAB) to tune both parameters as training proceeds. Experiments show that AdaFedSemi achieves a good balance between efficiency and accuracy by dynamically adjusting P and τ in different training phases.

DS-FL [42] tackles a similar problem to AdaFedSemi, where clients own labeled data while the server

owns unlabeled data. It proposes an ensemble pseudo-label solution to leverage the server-side unlabeled data. Specifically, instead of a single pseudo-label $\hat{y}_k$ for a data sample $\mathbf{X}_k$, it averages the pseudo-labels generated by all clients, i.e. $\hat{y}_k = \text{MEAN}_{c=1}^{C} g_{\theta_c}(\mathbf{X}_k)$. This creates an ensemble of client models and offers better performance. Moreover, as only pseudo-labels are transmitted instead of model parameters, the communication cost can be significantly saved. In addition, DS-FL observes that training on pseudo-labels leads to a high prediction entropy. It then proposes an entropy-reduced aggregation, which sharpens the local outputs $g_{\theta_c}(\mathbf{X}_k)$ before aggregation.

3.2 The Label-at-server Setting

The label-at-server setting, where clients do not have any labeled data, is more challenging than the label-at-client setting. The reason is that all clients own unlabeled data only and cannot provide additional supervision signals to the FL model. As shown in [35] and [34], training with only unlabeled data may lead to catastrophic forgetting of the knowledge learned from labeled data, and thus compromises the model performance.

To address the isolation between labeled data and unlabeled data, FedMatch [35] proposes a disjoint learning scheme that involves two sets of parameters for labeled and unlabeled data, respectively. The parameters for labeled data are fixed when training on unlabeled data, and vice versa, to prevent the knowledge from being overwritten. Disjoint learning brings additional benefits in communication efficiency, in that the parameters for unlabeled data, which are transmitted between participants and the server, are set to be sparse. In addition, to address the heterogeneous data held by different clients, FedMatch proposes an inter-client consistency loss, such that local models from different participants generate similar outputs on the same data.

SemiFL [34] takes another approach to solving the challenges. It proposes to fine-tune the global model with labeled data to enhance its quality and to alleviate the forgetting caused by unsupervised training at clients. Furthermore, instead of regularizing model outputs across clients, SemiFL proposes to maximize the consistency between client models and the global model. Specifically, the global model generates pseudo-labels for client-side unlabeled data, and the local models of clients are trained to fit the pseudo-labels. Empirical results show that SemiFL yields more competitive results than FedMatch.

4 Federated Self-supervised Learning

In this section, we introduce how self-supervised learning can be combined with FL to learn with decentralized and purely unlabeled data. Although there are two types of self-supervised learning, generative and contrastive learning, so far only contrastive methods have been studied in the FL setting, and thus we limit the discussions within federated contrastive self-supervised learning. The objective function can be formalized as

$$\min_{\theta} \frac{1}{C} \sum_{i=1}^{C} f_{ctr,i}(\theta), \text{s.t. } M_p(\theta) < \varepsilon_p, \tag{11}$$

where $f_{ctr,i}$ denotes f_{ctr} (Eqn. 7) evaluated at participant i. Compared to FL with full supervision, federated contrastive learning does not have globally consistent labels, and thus, the local contrastive objectives may deviate from one another to a greater extent. Therefore, heterogeneous data poses a greater challenge to federated contrastive learning. Table 5 summarizes existing works in federated contrastive self-supervised learning.

FedCA [49], as one of the earliest works to study federated self-supervised learning, proposes a dictionary module and an alignment module to solve the feature misalignment problem caused by data heterogeneity. Extending SimCLR, the dictionary module in FedCA aims to use the global model to generate consistent negative samples across clients, while the alignment module uses a set of public data to align the representations generated by local models. However, the alignment module of FedCA requires sharing a public dataset, which compromises data privacy.

Method	Label Isolation	Data Privacy	Data Heterogeneity	Efficiency Tradeoff
FedCA [49]	SimCLR	×	Dictionary & Alignment module	×
SSFL [52]	SimSiam	×	Personalized models	×
FedU [44]	BYOL	×	Selective divergence-aware update	×
FedEMA [50]	BYOL	×	Moving average client update	×
FedX [51]	Local relation loss	×	Global contrastive & relation loss	×
Orchestra [43]	Rotation prediction & clustering	Sending local centroids instead of all representations	Bi-level clustering	×

Table 5: Summary of techniques for federated self-supervised learning. × indicates that the proposed method does not focus on this issue.

SSFL [52] addresses the data heterogeneity problem in federated self-supervised learning with a personalized FL framework [53, 54], in which each participant trains s unique local model instead of training a shared global model. The drawback of SSFL is that the adopted self-supervised learning method requires a large batch size, which is hard to achieve on resource-limited edge devices.

FedU [44] designs a heterogeneity-aware aggregation scheme to address data heterogeneity in federated self-supervised learning. As discussed in Section 2.2.2, there are generally two networks in contrastive learning, an online network and a target network. Therefore, how to aggregate and update the two networks in FL with data heterogeneity becomes an important research question. With empirical experiments, FedU discovers that aggregating and updating only the online network yields better performances. Moreover, as FedU extends BYOL with an additional predictor model, it is also necessary to design an update rule for it. FedU designs a divergence-aware predictor update rule, which updates the local predictor only when its deviation from the global predictor is low. These rules ensure that data heterogeneity is well captured by local and global models.

Extending FedU, FedEMA [50] presents an extensive empirical study on the design components of federated contrastive learning. It performs experiments combining FL with MoCo, SimCLR, SimSiam, and BYOL, and identifies BYOL as the best base method. Based on the results, FedRMA with a divergence-aware moving average update rule is proposed. The difference between FedEMA and FedU is that, FedU overwrites the local online model with the global online model

$$\theta_{o,c}^{r} = \theta_{o}^{r-1}, \tag{12}$$

where $\theta_{o,c}^{r}$ denotes the local online model at client c and round r, and θ_{o}^{r-1} denotes the global online model aggregated at the previous round. On the contrary, FedEMA updates the local online model by interpolating between the global and local online models to adaptively incorporate global knowledge, i.e.

$$\theta_{o,c}^{r} = (1-\mu)\theta_{o}^{r-1} + \mu\theta_{o,c}^{r-1}, \tag{13}$$

where μ is a parameter based on weight divergence,

$$\mu = \min(\lambda\|\theta_{o,c}^{r-1} - \theta_{o}^{r-1}\|, 1). \tag{14}$$

While FedU and FedEMA are simple and effective, they both require *stateful* clients to keep track of the divergence between local and global models, and are thus not applicable in cross-device FL.

Contrary to FedU and FedEMA, Orchestra [43] proposes a theoretically guided federated self-supervised learning method that works with cross-device FL. Orchestra is based on the theory that feature representations with good clustering properties yield low classification errors. Therefore, in addition to contrastive learning, Orchestra aims to simultaneously enhance the clustering properties of all data representations. However, sharing all data representations for clustering may cause the problem of privacy leakage. Orchestra addresses the problem

with a bi-level clustering method, in which clients first cluster their data representations, and send only the local centroids to the server. The server performs a second clustering on the local centroids to obtain global clustering centroids, which are sent back to clients to compute cluster assignments. As local centroids reveal less information than all data representations, this bi-level clustering method better preserves data privacy.

Orthogonal to the above methods, FedX [51] proposes a versatile add-on module for federated self-supervised learning methods. FedX consists of both local and global relational loss terms that can be added to various contrastive learning modules. The local relational loss aims to ensure that under the local model, two augmentations of the same data sample have similar relations (similarities) to samples within the same batch $\mathcal{B}$, i.e.

$$\mathbf{r}_i^j = \frac{\exp(\text{sim}(\mathbf{z}_i, \mathbf{z}_j))}{\sum_{k\in\mathcal{B}} \exp(\text{sim}(\mathbf{z}_i, \mathbf{z}_k))}, \mathbf{r}_{i+}^j = \frac{\exp(\text{sim}(\mathbf{z}_{i+}, \mathbf{z}_j))}{\sum_{k\in\mathcal{B}} \exp(\text{sim}(\mathbf{z}_{i+}, \mathbf{z}_k))}, \tag{15}$$

$$L_{rel} = \text{JS}(\mathbf{r}_i, \mathbf{r}_{i+}), \tag{16}$$

where $\mathbf{z}_i, \mathbf{z}_{i+}$ denotes the feature representation of the i-th sample (with different augmentations) in the batch, $\mathbf{r}_i$ denotes the (normalized) similarity between the i-th sample and all other samples, and JS denotes the Jensen-Shannon divergence. The global relational loss is similarly defined such that under the local model, two augmentations of the same data sample have similar relations to the global representations. Empirical results show that FedX is versatile and can improve the performance of various contrastive learning methods in FL, including FedSimCLR, FedMoCo, and FedU.

5 Federated Transfer Learning

In this section, we summarize efforts that combine FL with transfer learning (FTL). We categorize existing works in FTL into homogeneous FTL and heterogeneous FTL, whose differences are introduced in Section 2.2.3.

5.1 Homogeneous FTL

In this section, we introduce research works on homogeneous FTL. Assuming that there are S source domains and T target domains, each of which is held by one participant, the objective of homogeneous FTL is as follows,

$$\min_{\theta_f,\theta_c} \sum_{i=1}^{S} f_{cls}(\mathcal{L}_i; \theta_f, \theta_c) + \sum_{i=1}^{S}\sum_{j=1}^{T} \lambda_{ij} f_{dom}(\mathcal{L}_i, \mathcal{U}_j; \theta_f), \text{s.t. } M_p(\theta_f, \theta_c) < \varepsilon_p, \tag{17}$$

where $\mathcal{L}_i, \mathcal{U}_j$ denote the labeled/unlabeled dataset held by the i-th source/target domain, respectively, f_{cls}, f_{dom} follow Eqn. 8, and λ_{ij} are hyperparameters used to select helpful source domains. If source domain i and target domain j are similar, we can assign a high λ_{ij}, and vice versa. Depending on how many source or target domains are involved, we can categorize existing works into two settings, single-source, and multi-source. In the multi-source setting, selecting the most appropriate source domain poses an additional challenge compared to the single-source setting. We introduce related works in both settings in the following sections.

5.1.1 Single-source Setting

The single-source setting in federated transfer learning commonly involves one server with labeled data, and multiple clients with unlabeled data. As the clients themselves may have different data distributions, each client creates a unique target domain, which requires a flexible adaptation method to tackle multiple targets.

To our knowledge, DualAdapt [56] is the first work to tackle the single-source, multi-target federated transfer learning problem. DualAdapt extends from the maximum classifier discrepancy (MCD) method [61]. Specifically, MCD involves a feature extractor θ_f and two classifiers θ_{c1}, θ_{c2}, trained with the following steps iteratively:

Method	Homo-geneous	# Source	# Target	Label Isolation	Data Privacy	Data Heterogeneity	Efficiency Tradeoff
DualAdapt [56]	✓	1	> 1	MCD & Pseudo-labeling& MixUp approximation	×	GMM weighting	×
FRuDA [58]	✓	1	> 1	DANN [33] & Optimal collaborator selection	×	Optimal collaborator selection	Lazy update
FADA [55]	✓	> 1	1	DANN [33] & Representation sharing	×	Gap statistics weighting	×
FADE [57]	✓	> 1	1	DANN	No representation sharing	CDAN Squared adversarial loss	×
EfficientFDA [60]	✓	> 1	1	Max. mean discrepancy (MMD)	Homomorphic encryption (HE)	×	Optimized HE operation
PrADA [86]	✓	2	1	Grouped DANN	Homomorphic encryption (HE)	×	×
SFTL [85]	×	1	1	Sample alignment loss	HE & Secret sharing (SS)	Sample alignment loss	×
SFHTL [84]	×	1	>1	Label propagation	Split learning [87]	Unified feature space	×

Table 6: Summary of techniques for (unsupervised) federated transfer learning. × indicates that the proposed method does not focus on this issue. 'Homogeneous' indicates whether the work focuses on homogeneous FTL (✓) or heterogeneous FTL (×). # Source, # Target denote the number of source and target domains considered in the work, respectively.

- First, $\theta_f, \theta_{c1}, \theta_{c2}$ are trained to minimize the error on the source domain $\mathcal{L}_1$.
- Second, given a target domain sample $\mathbf{X}_t$, we fix the feature extractor θ_f and maximize the discrepancy between the classifiers, i.e. $\max_{\theta_{c1},\theta_{c2}} L_{cd} = d(g_{\theta_f,\theta_{c1}}(\mathbf{X}_t), g_{\theta_f,\theta_{c2}}(\mathbf{X}_t))$. This step aims to find target samples that are dissimilar to the source domain.
- Third, the classifiers are fixed, and the feature extractor θ_f is trained to minimize L_{cd} to generate domain invariant features.

The FL setting creates two challenges for MCD. First, Step 2 should be taken at clients, yet as no labels are available, Step 2 may result in naive non-discriminative solutions. To address the problem, DualAdapt proposes client self-training, where pseudo-labels generated by the server model are used to train the classifiers in addition to L_{cd}. Second, to maintain a single feature extractor θ_f, Step 3 is done at the server, which has no access to target samples $\mathbf{X}_t$. DualAdapt proposes to use mixup [62] to approximate target samples $\mathbf{X}_t$. To further mitigate the impact of domain discrepancy, DualAdapt proposes to fit Gaussian mixture models (GMM) at each participant. At each participant, samples from other participants are re-weighed via the fit GMMs, such that impacts of highly dissimilar samples are mitigated.

FRuDA [58] proposes a system for single-source, multi-target federated transfer learning with DANN [33] Similar to DualAdapt, it also considers the setting with multiple unlabeled target domains, for which it proposes an optimal collaboration selection (OCS) method. The intuition of OCS is that, for a new target domain, instead of always transferring from the only source domain, it is also possible to transfer from an existing target domain that is closer to the new domain. To implement the intuition, OCS derives an upper bound for the transfer learning error from one domain to another,

$$\varepsilon_{CE,D_2}(h, h') \leq \theta_{CE}(\varepsilon_{L_1,D_1}(h, h') + 2\theta W(D_1, D_2)), \tag{18}$$

where $\varepsilon_{M,D}(h, l)$ denotes the error, measured by metric M, of the hypothesis h on data distribution D with the label function l, θ_{CE}, θ are constants, h, h' are source and target hypotheses, D_1, D_2 are source and target data distributions, respectively, and $W(D_1, D_2)$ denotes the Wasserstein distance between D_1, D_2. With Eqn. 18, the optimal collaborator of each target domain can be selected by minimizing the right-hand side. To further improve efficiency, a lazy update scheme, exchanging discriminator gradients every p iteration, is further proposed.

5.1.2 Multi-source Setting

A more challenging setting of federated transfer learning is the multi-source setting, where multiple source domains with labeled data are available to transfer knowledge to a single unlabeled target domain. In this setting,

it is necessary to select a source domain with helpful knowledge without directly observing source data.

To our knowledge, FADA [55] is the first work to tackle the multi-source federated transfer learning problem. FADA extends the adversarial domain adaptation [33] method, with a domain discriminator between each source domain and the target domain. The domain discriminator aims to tell whether each feature representation belongs to the source and the target domain, and the feature extractor is then trained to fool the domain discriminator to learn domain invariant features. To train the domain discriminator, FADA directly exchanges feature representations from both domains, which may lead to potential privacy threats. In addition, to select the most relevant source domain to transfer from, FADA proposes a source domain weighting method based on gap statistics. Gap statistics [63] measures how well the feature representations are clustered,

$$I = \sum_{r=1}^{k} \frac{1}{2n_r} \sum_{i,j \in C_r} \|\mathbf{z}_i - \mathbf{z}_j\|_2, \tag{19}$$

where $\mathbf{f}_i$ denotes the feature representation of the i-th sample, $C_1 \dots C_k$ denote the index set of k clusters, and n_r is the number of samples in cluster r. A low I indicates that the feature representations can be clustered with low intra-cluster variance, which usually indicates good features. FADA then computes how the gap statistics of the target domain drop after learning with each source domain, i.e.

$$I_i^{gain} = I_i^{r-1} - I_i^r, \tag{20}$$

where r denotes the communication round, and i denotes the source domain index. Finally, FADA applies weights on source domains via with $\mathrm{Softmax}(I_1^{gain}, I_2^{gain} \dots,)$.

FADE [57] improves over FADA by not sharing representations to learn the domain discriminator, thus better protecting data privacy. Instead, the domain discriminator is kept local at each client, and is trained locally and updated via parameter aggregation. FADE theoretically shows that the design leads to the same optimal values as FADA, but empirically leads to negative impacts. The issues of the design are that the trained discriminator may have low sensitivity (and thus takes longer to converge) and user mode collapse (and thus fail to represent heterogeneous data). To address the drawbacks, FADA presents two tricks. To tackle the low sensitivity issue, FADE squares the adversarial loss such that it is more reactive under large loss values. To tackle the user mode collapse issue, FADE proposes to maximize the mutual information between users (related to classes) and representations, and implements the idea with conditional adversarial domain adaptation (CDAN) [80].

EfficientFDA [60] is another improvement over FADA in that source and target domain feature representations are encrypted with homomorphic encryption (HE) [64], and the maximum mean discrepancy (MMD) [32] is computed over ciphertexts. As homomorphic encryption incurs large computation and communication costs, EfficientFDA further proposes two ciphertext optimizations. First, ciphertexts in each batch of samples are aggregated to reduce communication overhead. Second, for computing gradients with ciphertexts, the chain rule is applied to replace ciphertext computations with plaintexts to improve computational efficiency. Experiments show that EfficientFDA achieves privacy in federated transfer learning, while being 10-100x more efficient than naive HE-based implementations.

While the above works tackle the problem with multiple source domains with the same feature space, PrADA [86] tackles a different problem, involving two source domains with different feature spaces. PrADA considers a partially labeled target domain A $\{\mathbf{X}_l^A, y_l^A\} \cup \{\mathbf{X}_u^A\}$, a labeled source domain B $\{\mathbf{X}^B \in \mathbb{R}^{N_B \times D}, y^B\}$, and a feature source domain C $\{\mathbf{X}_C^A \in \mathbb{R}^{N_A \times D_C}\} \cup \{\mathbf{X}_C^B \in \mathbb{R}^{N_B \times D_C}\}$. Domains A and B share the same feature space with different distributions, while domain C aims to provide rich auxiliary features for samples in both A and B. PrADA presents a fine-grained domain adaptation technique, in which features from domain C are first manually grouped into g tightly relevant feature groups. Each feature group is then assigned a feature extractor and a domain discriminator to perform fine-grained, group-level domain adaptation. In addition, to protect data privacy, the whole training process is protected with homomorphic encryption. Experiments show that with the grouped domain adaptation, PrADA achieves better transferability and interpretability.

5.2 Heterogeneous FTL

In this section, we introduce existing works about heterogeneous FTL. Compared to homogeneous FTL, the main difference of heterogeneous FTL is that it commonly requires cross-domain links between data (e.g. different features of the same user ID, the same features from different users, etc.) to bridge the heterogeneous feature spaces. Formally, assuming a heterogeneous FTL setting with two parties, A and B, with data $\mathcal{D}_A, \mathcal{D}_B$, with $\mathcal{D}_{AB} = \mathcal{D}_A \cap \mathcal{D}_B$ being the overlapping dataset (i.e. cross-domain links), the objective of heterogeneous FTL is

$$\min_{\theta_A, \theta_B} L_A(\mathcal{D}_A; \theta_A) + L_B(\mathcal{D}_B; \theta_B) + \lambda L_{algn}(\mathcal{D}_{AB}; \theta_A, \theta_B), \text{s.t. } M_p(\theta_A, \theta_B) < \varepsilon_p, \tag{21}$$

where L_A, L_B are loss functions on dataset $\mathcal{D}_A, \mathcal{D}_B$, respectively, and L_{algn} is an alignment loss that aims to align the overlapping dataset $\mathcal{D}_{AB}$ between domains. However, in FL, sharing sample features or labels pose potential privacy threats. How to leverage the cross-domain sample links to transfer knowledge while preserving privacy thus becomes a key challenge to solve.

To our knowledge, SFTL [85] is the first work to tackle the heterogeneous FTL problem. It considers a two-party setting and assumes that some user IDs $\mathcal{I}_{AB}$ exist in both parties (with different features). SFTL proposes an alignment loss to minimize the difference between features of the same users to achieve knowledge transfer,

$$L_{algn} = \sum_{i \in \mathcal{I}_{AB}} d(g_{\theta_A}(\mathbf{X}_i^A), g_{\theta_B}(\mathbf{X}_i^B)), \tag{22}$$

where $\mathcal{I}_{AB}$ denotes the overlapping user ID set, $g_{\theta_A}, g_{\theta_B}$ denote neural network models of party A and B, and $\mathbf{X}_i^A, \mathbf{X}_i^B$ denote the features of user i held by party A and B, respectively. In addition, SFTL addresses the data privacy problem by designing two secure protocols for SFTL, one based on homomorphic encryption, and the other based on secret sharing (SS).

The drawbacks of SFTL are that it is limited to the two-party setting, and both A and B have only partial models and cannot perform independent inference. To address these drawbacks, SFHTL [84] proposes an improved framework that supports multiple parties. The main difficulty in the multi-party heterogeneous FTL is the lack of overlapping samples and labels. To address the lack of overlapping samples, SFHTL proposes a feature reconstruction technique to complement the missing non-overlapping features. Specifically, all parties are trained to project their features into a unified latent feature space. Then, each party learns a reconstruction function that projects the unified features to raw features. With the reconstruction functions, each party can expand the feature spaces of non-overlapping samples, thus enlarging the training dataset. In addition, SFHTL proposes a pseudo-labeling method based on label propagation [20] to address the lack of labels. Specifically, a nearest neighbor graph based on feature proximity in the unified feature space is constructed, and the labels are propagated from labeled samples to unlabeled samples via the graph. Finally, to protect the privacy of labels, SFHTL is trained with split learning, such that labels are not directly shared with other parties.

6 Datasets and Evaluations

Benchmarking datasets are important for the development of machine learning research. In this section, we introduce commonly used datasets and benchmarks for the problem of FL without full labels in the existing literature. A summary of datasets can be found in Table 7. We find out that for both federated semi-supervised and unsupervised learning, existing works mainly partition (e.g. according to Dirichlet distributions) datasets for centralized machine learning (e.g. CIFAR-10, CIFAR-100, SVHN) manually, and manually sample a subset of labels. On the contrary, for federated transfer learning, datasets generally form natural partitions (e.g. city in GTA5, product types in AmazonReview, etc.) based on different domains. We thus conclude that real-world datasets representing realistic data heterogeneity and label isolation problems are still needed to credibly evaluate federated semi-supervised and self-supervised methods.

Dataset	FL Methods without Full Labels			Application	# Domains	# Samples	Partition
	Semi	Self	Trans.				
CIFAR-10	✓[38, 39, 35, 40, 34]	✓[52, 44, 43, 51, 50]	×	CV	1	60000	Dirichlet & Uniform
CIFAR-100	✓[38, 34]	✓[43, 44, 50]	×	CV	1	60000	Dirichlet & Uniform
SVHN	✓[40, 34]	✓[51]	×	CV	1	73257	Dirichlet & Uniform
Sent140	✓[39]	×	×	NLP	1	1600498	Natural (Twitter User)
Reuters	✓[42]	×	×	NLP	1	11228	Dirichlet
IMDb	✓[42]	×	×	NLP	1	50000	Dirichlet
Landmark-23K	×	✓[52]	×	CV	1	1600000	Natural (Location)
Digit-Five	×	×	✓[55, 58]	CV	5	107348	Natural (Style)
Office-Caltech10	×	×	✓[55, 58, 60]	CV	4	2533	Natural (Style)
DomainNet	×	×	✓[55, 58]	CV	6	416401	Natural (Style)
AmazonReview	×	×	✓[55]	NLP	4	8000	Natural (Product Category)
Mic2Mic	×	×	✓[58]	Speech	4	65000	Natural (Device Type)
GTA5	×	×	✓[56]	CV	4	25000	Natural (Location)

Table 7: Commonly used datasets for evaluating FL methods without full labels. ✓and × indicate that the dataset has or has not been used for evaluating an FL setting without full labels, respectively. # domains, # samples denote the number of domains and the total number of samples in the dataset. Datasets with multiple domains are more commonly used for unsupervised federated transfer learning.

7 Related Surveys

Federated learning has attracted the attention of researchers worldwide. Therefore, there have been many survey papers that cover various aspects of FL. In this section, we summarize and analyze existing survey papers compared to our work. Table 8 shows a summary of comparisons between related surveys and ours.

First, our work differs from general surveys on FL [11, 10, 88] in that they provide comprehensive reviews on a wide range of FL aspects, including privacy preservation, communication reduction, straggler mitigation, incentive mechanisms, etc. Among them, communication and privacy are also important issues in the problem of FL without full labels and are covered in our survey. On the contrary, our survey is focused on a specific aspect, namely how to deal with unlabeled data. Second, our work also differs from surveys on semi-supervised learning [21], self-supervised learning [24], and transfer learning [22] in the centralized setting, in that while they extensively summarize machine learning techniques for unlabeled data, they fail to cover FL-specific challenges, such as label isolation, data privacy, etc. Finally, compared to surveys that focus on FL algorithms on non-i.i.d. data [89, 90, 91], our work focuses on leveraging unlabeled data to assist FL, while these surveys focus on FL with fully labeled data, but are not independent and identically distributed. Nonetheless, these surveys are related to our work in that non-i.i.d. data is an important challenge in all FL settings, and we also summarize how existing works address the challenge in the problem of FL without full labels.

The most related survey to our work is [59], which surveyed FL techniques to tackle data space, statistical, and system heterogeneity. Our work is similar to [59] in two ways. On one hand, statistical heterogeneity is a key challenge in FL, and we also summarize how existing works address the challenge in FL without full labels. On the other hand, homogeneous and heterogeneous FTL (Section 5) are powerful tools to solve statistical and data space heterogeneity, respectively, which are also covered in Sections 3 and 4 in [59]. Nonetheless, the main focus of [59] lies in supervised FL with labeled data, which is different from our work which additionally covers federated semi-supervised and self-supervised methods.

Survey Papers	Similarities	Differences
[10, 11, 88]	Similar to our survey, these papers review existing solutions to protect data privacy and reduce communication/computation overhead.	These papers cover a wide range of aspects in general FL, while our survey focuses on a specific problem of leveraging unlabeled data.
[22, 23, 24, 21, 92, 93]	Similar to our survey, these papers review machine learning methods for unlabeled data, including semi-supervised, self-supervised, and transfer learning.	These papers do not cover FL specific challenges, such as labeled data isolation, data heterogeneity, data privacy, etc.
[90, 89, 91]	Similar to our survey, these papers review methods in FL that address the problem of non-i.i.d. data (i.e. data heterogeneity).	These papers primarily focus on optimization algorithms for fully supervised FL, while our work focuses specifically on leveraging unlabeled data.
[59]	Similar to our survey, [59] covers methods to tackle data heterogeneity. Also, [59] reviews existing works on homogeneous and heterogeneous FTL.	[59] primarily focuses on heterogeneity in supervised FL, while our work focuses on leveraging unlabeled data and covers federated semi-supervised and self-supervised learning.

Table 8: Comparative analysis between our survey and related surveys.

Learning Paradigm	Main Techniques	Advantages	Disadvantages
Federated Semi-supervised Learning	Enhancing methods in centralized settings with 1. **Label isolation**: Pseudo-labeling, domain alignment, etc. 2. **Privacy**: DP, HE, etc. 3. **Data heterogeneity**: Source domain selection, divergence-aware update, etc. 4. **Efficiency tradeoff**: Sample selection, communication reduction, HE optimization, etc.	Similar formulation to conventional FL. Can directly perform supervised tasks.	Data heterogeneity inherently violates i.i.d. assumption. Large-scale unlabeled data creates an efficiency tradeoff.
Federated Self-supervised Learning		Full utilization of client data. Suitable for unsupervised tasks like retrieval, clustering, etc.	Data heterogeneity inherently violates i.i.d. assumption. Need labels for supervised tasks.
Federated Transfer Learning		Models data heterogeneity, which is a key challenge in FL. Flexible formulation (heterogeneous FTL).	Source domain selection requires intricate design or manual effort.

Table 9: A summary of techniques, advantages, and disadvantages of learning paradigms reviewed in this paper.

8 Conclusion and Future Directions

8.1 Summary of the Survey

In this paper, we present a survey about the problem of federated learning without full labels. We introduce three learning paradigms to solve the problem, federated semi-supervised learning, federated self-supervised learning, and federated transfer learning. We further review existing works in these paradigms and discuss how they address the crucial challenges, i.e. label isolation, privacy protection, data heterogeneity, and efficiency tradeoff. Table 9 shows a summary of the main techniques, advantages, and disadvantages of learning paradigms discussed in this paper. We finally present a summary of the datasets and benchmarks used to evaluate FL methods without full labels.

8.2 Future Directions

Compared to general FL with full supervision, the problem of FL without full labels is still under-explored. We highlight the following future directions in the context of FL without full labels.

8.2.1 Trustworthiness

Trustworthiness is an important aspect in real-world machine learning systems like FL. Generally speaking, users of machine learning systems would expect a system to be private, secure, robust, fair, and interpretable, which is what trustworthiness mean in the context of FL.

Unlabeled data can play an important role in enhancing trustworthiness from multiple aspects.

- **Robustness**: A robust system requires that its output should be insensitive to small noises added to the input. A machine learning system that is not robust can significantly compromise its security in real-world applications. For example, studies [69] have shown that is it possible to tweak physical objects to fool an object detection model. In applications like autonomous driving, this property becomes a security threat.

  Many research works have studied how to enhance robustness with unlabeled data [68, 67]. For example, Carmon et al. and Uesato et al. [67, 70] show that pseudo-labeling, one of the most common semi-supervised learning techniques, can boost the robustness by 3-5% over state-of-the-art defense models. Deng et al. [68] additionally find out that even out-of-distribution unlabeled data helps enhance robustness. Therefore, how these techniques can be adapted in the FL setting with heterogeneous data is an interesting future direction. Also, as common methods of learning robust models (i.e. adversarial training [81]) are inefficient, it is promising to study whether FL methods without full labels can be an efficient substitute.

- **Privacy**: In real-world machine learning applications, labeling data itself is a compromise of data privacy, as domain experts have to directly observe the data. Therefore, solving the FL problem without full labels inherently leads to better data privacy. In addition, unlabeled data provides a better way of navigating through the privacy-utility tradeoff in differential privacy (DP) [72]. For example, PATE [71] shows that with an additional set of unlabeled data, it simultaneously achieves a higher model accuracy and a tighter privacy bound compared to the state-of-the-art DPSGD method [73]. Therefore, how to select and leverage unlabeled data to aggregate client knowledge privately while maintaining good model accuracy is also a promising direction.

- **Interpretability**: Interpretability indicates that a machine learning system should be able to make sense of its decision, which generally creates trust between users and system developers. There are many ways to instill interpretability in machine learning, among which disentangled representation learning [74] is a popular direction. Informally speaking, disentangled representation aims to map the inputs to latent representations where high-level factors in the input data are organized in a structured manner in the

representations (e.g. brightness, human pose, facial expressions, etc.). Thus, disentangled representations provide intuitive ways to manipulate and understand deep learning models and features.

Much progress has been made in unsupervised disentangled representation learning. For example, InfoGAN [75] learns disentangled representations by maximizing the mutual information between the features and the output. Beta-VAE [76] disentangles features by adding an independence regularization on the feature groups. Therefore, it is promising to instill interpretability in FL via unlabeled data with disentangled representations. In FL, the participants commonly hold data with varying data distributions. Therefore, how to stably disentangle the heterogeneous feature distributions from multiple participants is a challenge for interpretable FL without full labels.

- **Fairness**: As machine learning models are increasingly involved in decision-making in the daily lives of people, the models should not discriminate one group of users against another (e.g. gender, race, etc.). Informally speaking, the fairness of a machine learning model g_θ over a sensitive attribute s can be described as the difference between the model performances given different values of s,

$$\Delta_{s,\theta} = \|m(g_\theta|s=1) - m(g_\theta|s=0)\|, \tag{23}$$

where $\|\cdot\|$ is a distance, and $m(g_\theta|s=1)$ is a performance metric stating how well the model performs when the sensitive attribute $s=1$.

When $m(g_\theta|s)$ does not involves labels (e.g. some groups have a higher probability to be predicted positive), FADE [57] provides a good solution to ensure group fairness. However, when $m(g_\theta|s=1)$ requires labeled data (e.g. classification accuracy is lower for under-represented groups), enforcing fairness with unlabeled data remains an open problem, both for general machine learning and FL.

8.2.2 Generalization to Unseen Domains

All the introduced techniques in this paper require at least observing the test domain such that it can work well on it. Even for federated transfer learning, some unlabeled samples in the target domain are still needed for successful adaptation. However, in real-world applications, it is often required to adapt to completely unseen domains. For example, FL models should try to adapt to new users that constantly join mobile applications, who, at the time of joining, have no interaction data available. The problem setting triggers research in federated domain generalization (FedDG). However, existing works in FedDG [77, 78] assume that all domains are fully labeled, which, as stated in this survey, is not realistic. It is thus important to study the FedDG problem under limited labeled data and large-scale unlabeled data.

8.2.3 Automatic FL without Full Labels

Automatic machine learning (AutoML) [82] is a class of methods that aim to achieve good model performances without manual tuning (e.g. architecture, hyperparameters, etc.). In FL without full labels, as different participants may hold heterogeneous labeled or unlabeled data, it may not be optimal for them to share the same model architecture. Integrating AutoML to FL without full labels thus enables participants to find personalized architectures to achieve the performance-efficiency tradeoff. However, participants with only unlabeled data cannot independently evaluate the performance of the model, creating challenges to automatic FL without full labels.

References

[1] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A Large-scale Hierarchical Image Database. 2009 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 248–255, 2009.

[2] Fabian Caba Heilbron, Victor Escorcia, Bernard Ghanem, and Juan Carlos Niebles. ActivityNet: A Large-scale Video Benchmark for Human Activity Understanding. 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 961–970, 2015.

[3] Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning Books and Movies: Towards Story-like Visual Explanations by Watching Movies and Reading Books. 2015 IEEE International Conference on Computer vision (ICCV), 19–27, 2015.

[4] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer Sentinel Mixture Models. International Conference on Learning Representations, 2017.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 770–778, 2016.

[6] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. 2017 IEEE International Conference on Computer Vision (ICCV), 2961–2969, 2017.

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT), 4171–4186, 2019.

[8] The European Parliament. General Data Protection Regulation. https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:02016R0679-20160504, 27 April 2016.

[9] State of California Department of Justice. California Consumer Privacy Act. https://oag.ca.gov/privacy/ccpa, 2018.

[10] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated Machine Learning: Concept and Applications. ACM Transactions on Intelligent Systems and Technology (TIST), 10.2 (2019): 1-19.

[11] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, and Others. Advances and Open Problems in Federated Learning. Foundations and Trends® in Machine Learning, 14.1–2 (2021), 1–210.

[12] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient Learning of Deep Networks from Decentralized Data. International Conference on Artificial Intelligence and Statistics (AISTATS), 1273–1282, 2017.

[13] 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 (MLSys), 429–450, 2020.

[14] Fan Lai, Xiangfeng Zhu, Harsha V. Madhyastha, and Mosharaf Chowdhury. Oort: Efficient Federated Learning via Guided Participant Selection. USENIX Symposium on Operating Systems Design and Implementation (OSDI), 19–35, 2021.

[15] Sai Praneeth Karimireddy, Satyen Kale, Mahryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. SCAFFOLD: Stochastic Controlled Averaging for Federated Learning. International Conference on Machine Learning (ICML), 5132–5143, 2020.

[16] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical Secure Aggregation for Privacy-preserving Machine Learning. ACM SIGSAC Conference on Computer and Communications Security (CCS), 1175–1191, 2017.

[17] Enmao Diao, Jie Ding, and Vahid Tarokh. HeteroFL: Computation and Communication Efficient Federated Learning for Heterogeneous Clients. International Conference on Learning Representations (ICLR), 2021.

[18] Chengliang Zhang, Suyi Li, Junzhe Xia, Wei Wang, Feng Yan, and Yang Liu. BatchCrypt: Efficient Homomorphic Encryption for Cross-Silo Federated Learning. USENIX Annual Technical Conference (ATC), 493–506, 2020.

[19] Amirhossein Reisizadeh, Aryan Mokhtari, Hamed Hassani, Ali Jadbabaie, and Ramtin Pedarsani. FedPAQ: A Communication-efficient Federated Learning Method with Periodic Averaging and Quantization. International Conference on Artificial Intelligence and Statistics (AISTATS), 2021–2031, 2020.

[20] Xiaojin Jerry Zhu. Semi-supervised Learning Literature Survey. University of Wisconsin-Madison, 2005.

[21] Olivier Chapelle, Bernhard Scholkopf, and Alexander Zien. Semi-supervised Learning. IEEE Transactions on Neural Networks, 20.3 (2009), 542–542.

[22] Sinno Jialin Pan and Qiang Yang. A Survey on Transfer Learning. IEEE Transactions on Knowledge Discovery and Data Engineering, 22.10 (2010), 1345–1359.

[23] Longlong Jing and Yingli Tian. Self-supervised Visual Feature Learning with Deep Neural Networks: A Survey. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 43.11 (2020), 4037–4058.

[24] Xiao Liu, Fanjin Zhang, Zhenyu Hou, Li Mian, Zhaoyu Wang, Jing Zhang, and Jie Tang. Self-supervised Learning: Generative or Contrastive. IEEE Transactions on Knowledge and Data Engineering (TKDE), 35.1 (2021), 857–876.

[25] Antti Tarvainen and Harri Valpola. Mean Teachers are Better Role Models: Weight-averaged Consistency Targets Improve Semi-supervised Deep Learning Results. Advances in Neural Information Processing Systems (NeurIPS), 2017.

[26] Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. Virtual Adversarial Training: A Regularization Method for Supervised and Semi-supervised Learning. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 41.8 (2018), 1979–1993.

[27] David Berthelot, Nicholas Carlini, Ian Goodfellow, Nicholas Papernot, Avital Oliver, and Colin A Raffel. MixMatch: A Holistic Approach to Semi-supervised Learning. Advances in Neural Information Processing Systems (NeurIPS), 2019.

[28] Carl Doersch, Abhinav Gupta, and Alexei A Efros. Unsupervised Visual Representation Learning by Context Prediction. IEEE International Conference on Computer Vision (ICCV), 1422–1430, 2015.

[29] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised Representation Learning by Predicting Image Rotations. International Conference on Learning Representations, 2018.

[30] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked Autoencoders are Scalable Vision Learners. IEEE/CVF Conference on Computer Vision and Pattern Recognition, 16000–16009, 2022.

[31] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How Transferable are Features in Deep Neural Networks?. Advances in Neural Information Processing Systems (NIPS), 2014.

[32] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael Jordan. Learning Transferable Features with Deep Adaptation Networks. International Conference on Machine Learning (ICML), 97–105, 2015.

[33] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial Training of Neural Networks. Journal of Machine Learning Research (JMLR), 17.1 (2016), 2096–2130.

[34] Enmao Diao, Jie Ding, and Vahid Tarokh. SemiFL: Semi-supervised Federated Learning for Unlabeled Clients with Alternate Training. Advances in Neural Information Processing Systems (NeurIPS), 2022.

[35] Wonyong Jeong, Jaehong Yoon, Eunho Yang, and Sung Ju Hwang. Federated Semi-supervised Learning with Inter-client Consistency & Disjoint Learning. International Conference on Learning Representations (ICLR), 2021.

[36] Jonas Geiping, Hartmut Bauermeister, Hannah Droöge, and Michael Moeller. Inverting Gradients-How Easy Is It to Break Privacy in Federated Learning? Advances in Neural Information Processing Systems (NeurIPS), 16937–16947, 2020.

[37] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. Federated Learning with Non-iid Data. arXiv preprint arXiv:1806.00582, 2018.

[38] Xiaoxiao Liang, Yiqun Lin, Huazhu Fu, Lei Zhu, and Xiaomeng Li. RSCFed: Random Sampling Consensus Federated Semi-supervised Learning. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 10154–10163, 2022.

[39] Chenyou Fan, Junjie Hu, and Jianwei Huang. Private Semi-supervised Federated Learning. International Joint Conference on Artificial Intelligence (IJCAI), 2009–2015, 2022.

[40] Lun Wang, Yang Xu, Hongli Xu, Jianchun Liu, Zhiyuan Wang, and Liusheng Huang. Enhancing Federated Learning with In-cloud Unlabeled Data. IEEE International Conference on Data Engineering (ICDE), 136–149, 2022.

[41] Xinyang Lin, Hanting Chen, Yixing Xu, Chao Xu, Xiaolin Gui, Yiping Deng, and Yunhe Wang. Federated Learning with Positive and Unlabeled Data. International Conference on Machine Learning (ICML), 13344–13355, 2022.

[42] Sohei Itahara, Takayuki Nishio, Yusuke Koda, Masahiro Morikura, and Koji Yamamoto. Distillation-based Semi-supervised Federated Learning for Communication-efficient Collaborative Training with Non-iid Private Data. IEEE Transactions on Mobile Computing (TMC), 22.1 (2021), 191–205.

[43] Ekdeep Lubana, Chi Ian Tang, Fahim Kawsar, Robert Dick, and Akhil Mathur. Orchestra: Unsupervised Federated Learning via Globally Consistent Clustering. International Conference on Machine Learning (ICML), 14461–14484, 2022.

[44] Weiming Zhuang, Xin Gan, Yonggang Wen, Shuai Zhang, and Shuai Yi. Collaborative Unsupervised Visual Representation Learning from Decentralized Data. IEEE/CVF International Conference on Computer Vision (ICCV), 4912–4921, 2021.

[45] Ting Chen, Simon Kornblith, Mohammad Norouzi, Geoffrey Hinton. A Simple Framework for Contrastive Learning of Visual Representations. International Conference on Machine Learning (ICML), 1597–1607, 2020.

[46] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum Contrast for Unsupervised Visual Representation Learning. IEEE/CVF Conference on Computer Vision and Pattern Recognition, 9729–9738, 2020.

[47] Xinlei Chen and Kaiming He, Exploring Simple Siamese Representation Learning. IEEE/CVF Conference on Computer Vision and Pattern Recognition, 15750–15758, 2021.

[48] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernado Avila Pires, and Others. Bootstrap Your Own Latent-A New Approach to Self-supervised Learning. Advances in Neural Information Processing Systems, 21271–21284, 2020.

[49] Fengda Zhang, Kun Kuang, Zhaoyang You, Tao Shen, Jun Xiao, Yin Zhang, Chao Wu, Yueting Zhuang, and Xiaolin Li. Federated Unsupervised Representation Learning. arXiv prepring arXiv:2010.08982, 2020.

[50] Weiming Zhuang, Yonggang Wen, and Shuai Zhang. Divergence-aware Federated Self-supervised Learning. International Conference on Learning Representations (ICLR), 2022.

[51] Sungwon Han, Sungwon Park, Fangzhao Wu, Sundong Kim, Chuhan Wu, Xing Xie, and Meeyoung Cha. FedX: Unsupervised Federated Learning with Cross Knowledge Distillation. European Conference on Computer Vision (ECCV), 691–707, 2022.

[52] Chaoyang He, Zhengyu Yang, Erum Mushtaq, Sunwoo Lee, Mahdi Soltanolkotabi, and Salman Avestimehr. SSFL: Tackling Label Deficiency in Federated Learning via Personalized Self-supervision. arXiv preprint arXiv:2110.02470, 2021.

[53] Alysa Ziying Tan, Han Yu, Lizhen Cui, and Qiang Yang. Towards Personalized Federated Learning. IEEE Transactions on Neural Networks and Learning Systems, Early Access, 2022.

[54] Tian Li, Shengyuan Hu, Ahmad Beirami, and Virginia Smith. Ditto: Fair and Robust Federated Learning through Personalization. International Conference on Machine Learning (ICML), 6357–6368, 2021.

[55] Xingchao Peng, Zijun Huang, Yizhe Zhu, and Kate Saenko. Federated Adversarial Domain Adaptation. International Conference on Learning Representations (ICLR), 2020.

[56] Chun-Han Yao, Boqing Gong, Hang Qi, Yin Cui, Yukun Zhu, and Ming-Hsuan Yang. Federated Multi-Target Domain Adaptation. IEEE/CVF Winter Conference on Applications of Computer Vision (WACV), 1424–1433, 2022.

[57] Junyuan Hong, Zhuangdi Zhu, Shuyang Yu, Zhangyang Wang, Hiroko H. Dodge, and Jiayu Zhou. Federated Adversarial Debiasing for Fair and Transferable Representations. ACM SIGKDD Conference on Knowledge Discovery & Data Mining (KDD), 617–627, 2021.

[58] Shaoduo Gan, Akhil Mathur, Anton Isopoussu, Fahim Kawsar, Nadia Berthouze, and Nicholas D. Lane. FRuDA: Framework for Distributed Adversarial Domain Adaptation. IEEE Transactions on Parallel and Distributed Systems (TPDS), 33.11 (2022), 3153–3164.

[59] Dashan Gao, Xin Yao, and Qiang Yang. A Survey on Heterogeneous Federated Transfer Learning. arXiv preprint arXiv:2210.04505, 2022.

[60] Hua Kang, Zhiyang Li, and Qian Zhang. Communicational and Computational Efficient Federated Domain Adaptation. IEEE Transactions on Parallel and Distributed Systems (TPDS), 33.12 (2022), 3678–3689.

[61] Kuniaki Saito, Kohei Watanabe, Yoshitaka Ushiku, and Tatsuya Harada. Maximum Classifier Discrepancy for Unsupervised Domain Adaptation. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 3723–3732, 2018.

[62] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. MixUp: Beyond Empirical Risk Minimization. International Conference on Learning Representations (ICLR), 2017.

[63] Robert Tibshirani, Guenther Walther, and Trevor Hastie. Estimating the Number of Clusters in a Data Set via the Gap Statistic. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 63.2 (2001), 411–423.

[64] Yoshinori Aono, Takuya Hayashi, Lihua Wang, Shiho Moriai, and Others. Privacy-preserving Deep Learning via Additively Homomorphic Encryption. IEEE Transactions on Information Forensics and Security, 13.5 (2017), 1333–1345.

[65] Fan Lai, Yinwei Dai, Sanjay Singapuram, Jiachen Liu, Xiangfeng Zhu, Harsha Madhyastha, and Mosharaf Chowdhury. FedScale: Benchmarking Model and System Performance of Federated Learning at Scale. International Conference on Machine Learning, 11814–11827, 2022.

[66] Sebastian Caldas, Sai Meher Karthik Duddu, Peter Wu, Tian Li, Jakub Konečnỳ, H. Brendan McMahan, Virginia Smith, and Ameet Talwalkar. LEAF: A Benchmark for Federated Settings arXiv preprint arXiv:1812.01097, 2018.

[67] Yair Carmon, Aditi Raghunathan, Ludwig Schmidt, John C. Duchi, and Percy S. Liang. Unlabeled Data Improves Adversarial Robustness. Advances in Neural Information Processing Systems (NeurIPS), 2019.

[68] Zhun Deng, Linjun Zhang, Amirata Ghorbani, and James Zou. Improving Adversarial Robustness via Unlabeled Out-of-domain Data. International Conference on Artificial Intelligence and Statistics (AISTATS), 2845–2853, 2021.

[69] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial Examples in the Physical World. Artificial Intelligence Safety and Security, 99–112, 2018.

[70] Jean-Baptiste Alayrac, Jonathan Uesato, Po-Sen Huang, Alhussein Fawzi, Robert Stanforth, and Pushmeet Kohli. Are Labels Required for Improving Adversarial Robustness? Advances in Neural Information Processing Systems (NeurIPS), 2019.

[71] Nicholas Papernot, Martin Abadi, Ulfar Erlingsson, Ian Goodfellow, and Kunal Talwar. Semi-supervised Knowledge Transfer for Deep Learning from Private Training Data. Internation Conference on Learning Representations (ICLR), 2017.

[72] Cynthia Dwork and Aaron Roth. The Algorithmic Foundations of Differential Privacy. Foundations and Trends®in Theoretical Computer Science, 9.3–4 (2014), 211–407.

[73] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep Learning with Differential Privacy. ACM SIGSAC Conference on Computer and Communications Security (CCS), 308–318, 2016.

[74] Hsin-Ying Lee, Hung-Yu Tseng, Jia-Bin Huang, Maneesh Singh, and Ming-Hsuan Yang. Diverse Image-to-image Translation via Disentangled Representations. European Conference on Computer Vision (ECCV), 35–51, 2018.

[75] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets. Advances in Neural Information Processing Systems (NeurIPS), 2016.

[76] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. Beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework. International Conference on Learning Representations (ICLR), 2017.

[77] Quande Liu, Cheng Chen, Jing Qin, Qi Dou, and Pheng-Ann Heng. FedDG: Federated Domain Generalization on Medical Image Segmentation via Episodic Learning in Continuous Frequency Space. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 1013–1023, 2021.

[78] A. Tuan Nguyen, Philip Torr, and Ser-Nam Lim. FedSR: A Simple and Effective Domain Generalization Method for Federated Learning. Advances in Neural Information Processing Systems (NeurIPS), 2022.

[79] Dong-Hyun Lee. Pseudo-Label: The Simple and Efficient Semi-supervised Learning Method for Deep Neural Networks. Workshop on Challenges in Representation Learning, ICML, 2013.

[80] Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Michael I. Jordan. Conditional Adversarial Domain Adaptation. Advances in Neural Information Processing System (NIPS), 2018.

[81] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards Deep Learning Models Resistant to Adversarial Attacks. International Conference on Learning Representations (ICLR), 2018.

[82] Xin He, Kaiyong Zhao, Xiaowen Chu. AutoML: A Survey of the State-of-the-art. Knowledge-Based Systems, 212 (2021), 106622.

[83] Michael T. Rosenstein, Zvika Marx, Leslie Pack Kaelbling, and Thomas G. Dietterich. To Transfer or Not to Transfer. NIPS 2005 Workshop on Transfer Learning, 2005.

[84] Siwei Feng, Boyang Li, Han Yu, Yang Liu, and Qiang Yang. Semi-Supervised Federated Heterogeneous Transfer Learning. Knowledge-Based Systems, 252 (2022), 109384.

[85] Yang Liu, Yan Kang, Chaoping Xing, Tianjian Chen, and Qiang Yang. A Secure Federated Transfer Learning Framework. IEEE Intelligent Systems, 35.4 (2020), 70–82.

[86] Yan Kang, Yuanqin He, Jiahuan Luo, Tao Fan, Yang Liu, and Qiang Yang. Privacy-Preserving Federated Adversarial Domain Adaptation over Feature Groups for Interpretability. IEEE Transactions on Big Data, 2022.

[87] Praneeth Vepakomma, Otkrist Gupta, Tristan Swedish, and Ramesh Raskar. Split Learning for Health: Distributed Deep Learning Without Sharing Raw Patient Data. arXiv preprint arXiv:1812.00564, 2018.

[88] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated Learning: Challenges, Methods, and Future Directions. IEEE Signal Processing Magazine, 37.3 (2020), 50–60.

[89] Jianyu Wang, Zachary Charles, Zheng Xu, Gauri Joshi, H Brendan McMahan, Maruan Al-Shedivat, and Others. A Field Guide to Federated Optimization. arXiv preprint arXiv:2107.06917, 2021.

[90] Qinbin Li, Yiqun Diao, Quan Chen, and Bingsheng He. Federated Learning on Non-iid Data Silos: An Experimental Study. IEEE International Conference on Data Engineering (ICDE), 965–978, 2022.

[91] Hangyu Zhu, Jinjin Xu, Shiqing Liu, and Yaochu Jin. Federated Learning on Non-iid Data: A Survey. Neurocomputing, 465 (2021), 371–390.

[92] Jesper E. Van Engelen and Holger H. Hoos. A Survey on Semi-supervised Learning. Machine Learning, 109.2 (2020), 373–440.

[93] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A Comprehensive Survey on Transfer Learning. Proceedings of the IEEE, 109.1 (2020), 43–76.

FedCLIP: Fast Generalization and Personalization for CLIP in Federated Learning

Wang Lu[1] Xixu Hu[2] Jindong Wang[3*] Xing Xie[3]
[1] National Engineering Research Center, Beijing, China
[2] City University of Hong Kong, Hong Kong
[3] Microsoft Research Asia, Beijing, China
newlw230630@gmail.com, xixuhu2-c@my.cityu.edu.hk, {jindong.wang, xingx}@microsoft.com

Abstract

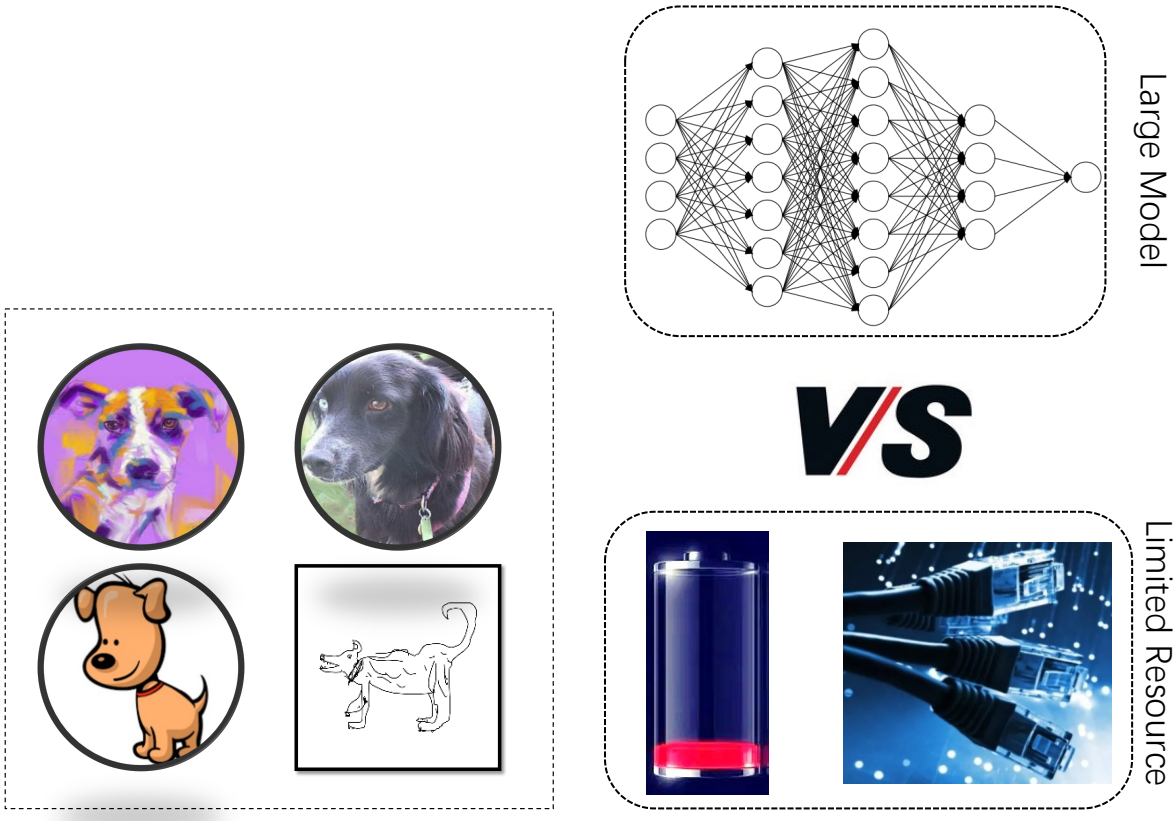
*Federated learning (FL) has emerged as a new paradigm for privacy-preserving computation in recent years. Unfortunately, FL faces two critical challenges that hinder its actual performance: data distribution heterogeneity and high resource costs brought by large foundation models. Specifically, the non-IID data in different clients make existing FL algorithms hard to converge while the high resource costs, including computational and communication costs that increase the deployment difficulty in real-world scenarios. In this paper, we propose an effective yet simple method, named FedCLIP, to achieve fast generalization and personalization for CLIP in federated learning. Concretely, we design an attention-based adapter for the large model, CLIP, and the rest operations merely depend on adapters. Lightweight adapters can make the most use of pretrained model information and ensure models be adaptive for clients in specific tasks. Simultaneously, small-scale operations can mitigate the computational burden and communication burden caused by large models. Extensive experiments are conducted on three datasets with distribution shifts. Qualitative and quantitative results demonstrate that FedCLIP significantly outperforms other baselines (**9**% overall improvements on PACS) and effectively reduces computational and communication costs (**283x** faster than FedAVG). Our code will be available at:* `https://github.com/microsoft/PersonalizedFL`.

1 Introduction

The success of machine learning, especially deep learning, is inseparable from a large amount of data. However, data, as an important resource, usually scatter across different individuals or organizations. In recent years, people pay more attention to data privacy and security and some organizations even enact relevant regulations and laws, e.g. The EU general data protection regulation (GDPR) [49] and China's cyber power [20]. Under this circumstance, direct raw data communication can be impossible in reality, making traditional data-centric machine learning paradigms unlikely to work. To cope with this challenge, federated learning (FL) [53] emerges as a new distributed machine learning paradigm and has been widely adopted in various applications.

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

*Corresponding author: Jindong Wang: jindong.wang@microsoft.com.



(a) Data distribution shifts (b) Huge resource demands

Figure 1: Existing issues in federated learning. In Figure 1(a), circles denote participated clients while squares denote unseen targets.

Federated learning makes it possible to perform model aggregation without directly accessing the raw user data from different clients. One of the earliest works in FL is called FedAVG [33] which aggregates distributed information using a simple and powerful averaging algorithm. FedAVG mainly contains four steps, including training local models with local data, uploading local models to the server, aggregating models in the server, and distributing the aggregated model to each individual or organization. These four steps are executed for multiple rounds for better information aggregation. FedAVG can ensure that raw data does not leave the local client and thus protect data privacy and security. Due to its simplicity and great performance, FedAVG quickly became popular in many areas [24, 40, 3].

In this paper, we are specially interested in federated learning under the large foundation models era [4]. Foundation models, as suggested by the name, have become increasingly popular in different machine learning tasks, such as Vision Transformer in computer vision [56] and the GPT series in natural language processing [37]. Since these models are extremely large, e.g., GPT-3 [5] has 175 billion parameters, our key question is: how to perform effective and efficient federated learning using these large models?

Specifically, two critical research challenges arise in this situation: data distribution shifts and huge resource demands. On the one hand, data distribution shifts widely exist in the real world, e.g. figures shown in Figure 1(a). When meeting heterogeneous data, common federated learning methods can suffer from slow convergence and low accuracy due to inconsistent optimization directions, local optima, or some other factors [12]. A qualified FL model can cope with both various clients and unseen targets, i.e. personalization and generalization. On

the other hand, huge resource demands of increasingly popular large models lead to conflicts with realistically constrained resources, as shown in Figure 1(b). In addition to high computational costs, communication cost is also a critical metric in federated learning. For instance, the CLIP [36] model based on VIT-B/32 contains more than 10^8 trainable parameters and most existing networks cannot afford to transmit it quickly. Achieving fast generalization and personalization with minimal resource costs is an urgent issue to be addressed.

Some existing work tried to address the issues mentioned above [32, 55, 14]. FedAP [32] attempted to learn the similarity among clients and then leveraged the learned similarity matrix to guide aggregation. FedAP could achieve acceptable personalization results but it ignored generalization. Another paper [55] discussed two gaps, including the out-of-sample gap and the participation gap. These two gaps correspond to goals of generalization and personalization respectively. This paper performed extensive empirical studies to analyze these issues but it did not offer a possible solution for large models. PromptFL [14] only updated the prompts instead of the whole model to accelerate the whole process. However, clients still require large amounts of computation and PromptFL is not designed for personalization and generalization.

In this paper, we propose FedCLIP to achieve fast generalization and personalization for CLIP in federated learning. Since larger pretrained models, e.g. CLIP, have contained enough prior information, our goal is to find where we should focus in specific tasks. The core part of FedCLIP is AttAI, an attention-based adapter for the image encoder in CLIP. Instead of finetuning whole networks, AttAI directly utilizes fixed features extracted by pretrained models and explores where FedCLIP should pay attention to for specific tasks. Simply training AttAI can ensure FedCLIP preserving prior information as much as possible while it allows models adapted for specific tasks. Through AttAI, FedCLIP does not rely on pretrained models anymore once obtaining diversified and robust features and thus FedCLIP can save large amounts of computational costs and communication costs. Therefore, FedCLIP is extensible and can be deployed to many applications.

Our contributions are as follows.

1. We propose FedCLIP, a fast generalization and personalization learning method for CLIP in federated learning. It can achieve personalization for participating clients and its remarkable generalization ability can attract new clients.

2. Extensive experiments on three public image benchmarks demonstrate that FedCLIP can have achieved personalization and generalization performance at the same time (**9**% overall improvements on PACS). More importantly, FedCLIP reduces the number of trainable parameters thus saving communication costs and computational costs (**283x** faster than FedAVG).

3. FedCLIP is extensible and can be applied in many real applications, which means it can work well in many circumstances. We can even embed it in some other architectures, e.g. BERT [46] and ViT [16]. Our code will be available at: `https://github.com/microsoft/PersonalizedFL`.

The remainder of this paper is organized as follows. In Sec. 2, we introduce related work. And then we elaborate on the proposed method in Sec. 4. Extensive experiments are reported and analyzed in Sec. 4. Finally, we conclude the paper and provide possible future work in Sec. 5.

2 Related Work

2.1 Challenges in Machine Learning

Machine learning has achieved great success and gradually entered people's daily lives [42, 34, 50]. It has been applied to many fields, e.g. human activity recognition [29], face recognition [28], and healthcare [8]. Successful machine learning applications, especially deep learning based applications, often require a large amount of data and lots of computational resources. In most cases, a deluge of data and computing resources can lead to easy

success, such as ChatGPT [47]. However, data and computation also mean money and resources. In reality, it is impossible to aggregate all data together in some situations. There seems to be a contradiction between the massive resource requirements of traditional methods and the limited real environment.

Generalization is another challenging problem caused by data distribution shifts. Its goal is to learn a generalized model with limited data and it expects that the learned model can work well on unseen targets with unknown distributions. [51] gives a survey on domain generalization and first groups existing methods into three categories, including data manipulation [31], representation learning [30], and learning strategy [19].

2.2 Federated Learning

Data is often scatted everywhere and cannot be aggregated together due to some factors, such as laws and regulations [49] and the awakening of people's awareness of data security and privacy protection. In such an environment, federated learning came into being [53, 41]. According to [53], federated learning can be grouped into three categories, including horizontal federated learning, vertical federated learning, and federated transfer learning. Most deep learning based methods belong to horizontal federated learning and so is this paper. For a more detailed introduction, please refer to the survey [27].

FedAVG is a traditional horizontal federated learning method [33]. Although it is simple, it was applied in many applications. When meeting data distribution heterogeneity, FedAVG appeared powerless [43]. And many researchers proposed various methods to solve the above problems. FedProx [25] added a proximal regularized term to FedAVG and it allowed slight model gaps between clients and the server. In FedBN [26], the authors thought that parameters in batch normalization layers can represent data distribution, and keeping specific batch normalization layers for each client could make local models personalized. Another latest method, FedAP [32], learned the similarity between clients based on the statistics of the batch normalization layers while preserving the specificity of each client with different local batch normalization. The above methods all achieve satisfactory results in their corresponding scenarios. However, most of them focused on personalization and ignored generalization issues [7].

Generalization in federated learning is a novel problem. In recent two years, some papers tried to solve this problem. [55] first discussed the generalization in federated learning and it proposed a framework to disentangle performance gaps, including out-of-sample gaps and participation gaps. FED-DRO [7] proposed a novel federated learning framework to explicitly decouple a model's dual duties with two prediction tasks and it mainly focused on label shifts. Some other work tried to adapt existing domain generalization methods to generalization in federated learning [15, 45, 35, 6]. FL Games [15] utilized Nash equilibrium to learn causal features that were invariant across clients which is similar to Invariant Risk Minimization (IRM) [2]. FedSAM [35] proposed a general effective algorithm based on Sharpness Aware Minimization (SAM) local optimizer [11]. Although these methods can bring generalization, they were not designed for large models and could not make full use of knowledge brought by pretrained models.

2.3 CLIP and Large Models

From perceptron [13] to AlexNet [1] to ResNet [17] to Vision transformer [56] to CLIP [36], pretrained models have become larger and larger. The importance of pretrained models has been increasing and pretrained models contain a growing amount of knowledge. For specific applications, researchers usually choose suitable backbone models and then adopt some techniques, e.g. finetune [44], to slightly adapt pretrained models. Since pretrained models are trained via a large amount of data, features extracted from them are often generalized and insightful. Few works pay attention to large models in federated learning and high demands of computational costs and communication costs hinder the development of this field. In this paper, we focus on CLIP in federated learning.

CLIP [36] learned SOTA image representations from scratch on a dataset of 400 million(image, text) pairs collected from the internet. The natural language was used to reference learned visual concepts. It has been

applied in many fields and demonstrated its superiority [38, 22]. However, in federated learning, CLIP is still in its infancy. PromptFL [14] replaced the federated model training with the federated prompt training to simultaneously achieve efficient global aggregation and local training by exploiting the power of foundation models in a distributed way. However, it still requires certain computational costs and it is not designed for data distribution heterogeneity problems. Moreover, it is hard to tune the hyperparameters for the prompt techniques in Transformer.In this paper, we focus on fast personalization and generalization for CLIP.

3 Method

3.1 Problem Formulation

In a generalization and personalization federated learning setting, N different clients, denote as $\{C_1, C_2, \cdots, C_N\}$, participate in exchanging information and they have data, denoted as $\{\mathcal{D}_1, \mathcal{D}_2, \cdots, \mathcal{D}_N\}$ with different distributions, which means $P(\mathcal{D}_i) \neq P(\mathcal{D}_j)$. In this paper, we only focus on homogeneous data with the same input space and output space, i.e. $\mathcal{X}_i = \mathcal{X}_j, \mathcal{Y}_i = \mathcal{Y}_j, \forall i \neq j$. Each dataset, $\mathcal{D}_i = \{(\mathbf{x}_{i,j}, y_{i,j})\}_{j=1}^{n_i}$, consists of three parts, a training dataset $\mathcal{D}_i^{train} = \{(\mathbf{x}_{i,j}^{train}, y_{i,j}^{train})\}_{j=1}^{n_i^{train}}$, a validation dataset $\mathcal{D}_i^{valid} = \{(\mathbf{x}_{i,j}^{valid}, y_{i,j}^{valid})\}_{j=1}^{n_i^{valid}}$ and a test dataset $\mathcal{D}_i^{test} = \{(\mathbf{x}_{i,j}^{test}, y_{i,j}^{test})\}_{j=1}^{n_i^{test}}$. Three sub-datasets in each client have no overlap and $n_i = n_i^{train} + n_i^{valid} + n_i^{test}, \mathcal{D}_i = \mathcal{D}_i^{train} \cup \mathcal{D}_i^{valid} \cup \mathcal{D}_i^{test}$. Our goal is to aggregate all clients' information with preserving data privacy and security and learn a good model f for each client $\mathcal{D}_i$:

$$\min_f \frac{1}{N} \sum_{i=1}^{N} \frac{1}{n_i^{test}} \sum_{j=1}^{n_i^{test}} \ell(f(\mathbf{x}_{i,j}^{test}), y_{i,j}^{test}), \tag{24}$$

where ℓ is a loss function. Moreover, for generalization, we assume that there exist M different clients, denote as $\{F_1, F_2, \cdots, F_M\}$, with data $\{\mathcal{D}_1^F = \{(\mathbf{x}_{i,j}, y_{i,j})\}_{j=1}^{m_1}, \mathcal{D}_2^F = \{(\mathbf{x}_{i,j}, y_{i,j})\}_{j=1}^{m_2}, \cdots, \mathcal{D}_N^F = \{(\mathbf{x}_{i,j}, y_{i,j})\}_{j=1}^{m_M}\}$. These M clients do not participate in training, and we hope f can also be able to perform well on these clients.

$$\min_f \frac{1}{M} \sum_{i=1}^{M} \frac{1}{m_i} \sum_{j=1}^{m_i} \ell(f(\mathbf{x}_{i,j}), y_{i,j}), \tag{25}$$

3.2 Preliminaries

CLIP CLIP, Contrastive Language Image Pre-training, is an efficient and scalable method of learning [36]. To compensate for the problems caused by the amount of data and model parameters, it trained a large model with over 4×10^8 pairs of data. With help of natural language supervision, CLIP can better understand concepts of visual images and better learn the semantic connections behind images. Usually, CLIP models contain more information and they might be more robust.

A simple CLIP model regularly contains two parts, an image encoder f^I and a text encoder f^T. In common models, labels are frequently represented as numbers or one-hot vectors. For CLIP, to better utilize semantic information, these labels are often transformed into sentences, e.g. 'A photo of dogs'. And then text feature vectors, $\mathbf{T}$ are extracted from these sentences via f^T. Concurrently, images are encoded into visual feature vectors, $\mathbf{I}$, via f^I. Cosine similarities between $\mathbf{T}$ and $\mathbf{I}$ are used to training and predicting.

FedAVG In FedAVG [33], each client trains f with local clients' data, and then parameters of updated models, w_i, are transmitted to the server. The server typically aggregates the parameters according to Eq. 26,

$$w^* = \sum_{i=1}^{N} \frac{n_i}{\sum_{j=1}^{N} n_j} w_i \tag{26}$$

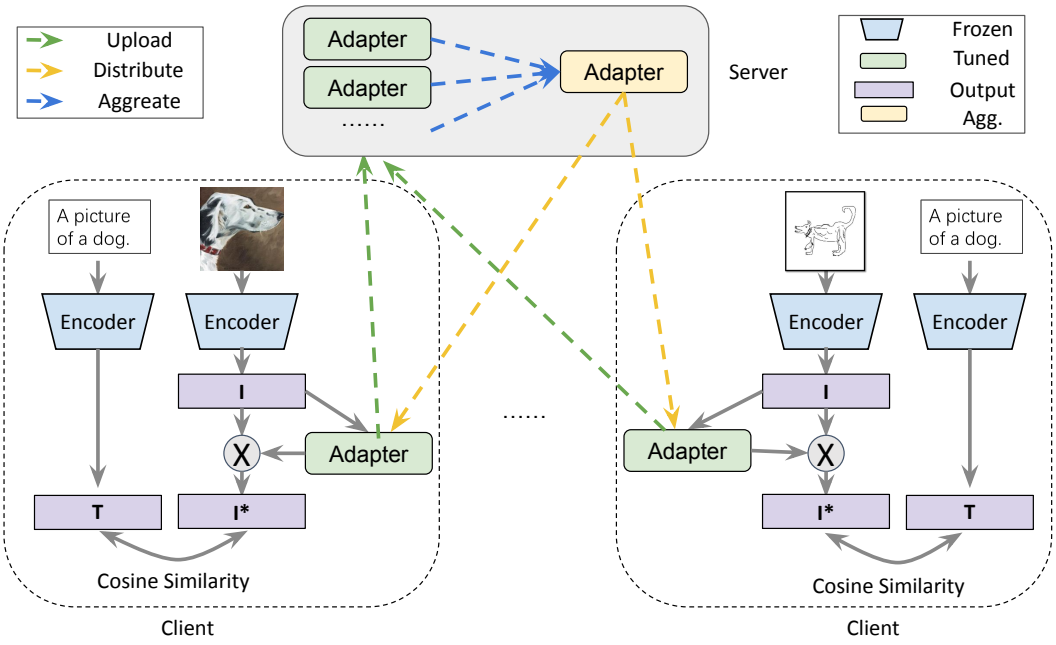


Figure 2: The framework of FedCLIP.

After aggregation, w^* is distributed. When $|w|$ is larger, the server cannot afford communication costs.

3.3 FedCLIP

To reduce computational costs and communications and make the most use of existing pretrained model information, we propose FedCLIP. Pretrained models already have abilities to extract robust and diversified features. Tuning whole networks with limited data can compromise the original ability of pretrained models. What we need to do is to try our best to preserve useful prior knowledge and let it be used to a suitable extent for our task. Besides, tuning large networks is impractical in federated learning due to limited resources in reality. Therefore, instead of operating on the whole model, FedCLIP concentrates on a simple attention-based adapter for the image encoder, AttAI.

Figure 2 gives the framework of FedCLIP. As shown in Figure 2, our method mainly contains four steps.

1. For Client i, we utilize a pretrained CLIP model to extract features of data, denoted as T_i and I_i.
2. In each client, we utilize $\mathcal{D}_i^{train}$ to train the corresponding adapter, g_i. And then we upload $\{g_i\}_{i=1}^N$ to the server.
3. In the server, the parameters of all g_i are weighted averaged and we can obtain g^*. The server then distributes g^* to each client and updates the parameters of each g_i.
4. Repeat Step 2 and Step 3 until convergence or reaching maximum rounds.

In step 1, we utilize the pretrained CLIP model to extract features. We consider the pretrained model is so powerful that we do not need to explore some other features. For $(\mathbf{x}, y)$, we can obtain corresponding features,

$$\mathbf{I} = f^I(\mathbf{x}), \mathbf{T} = f^T(y) \tag{27}$$

What we need to do next is to identify which parts of features are suitable for our specific tasks. Therefore, we introduce an attention-based adapter, g, to locate where we should concentrate on. Particularly, we utilize one linear layer, Tahn activation function, one linear layer, and Softmax activation function to construct g. The Softmax function is used to ensure our final outputs ranging from 0 to 1. Once we obtain the attention vector $att = g(\mathbf{I})$, we utilize it to update the visual feature via a dot multiply operation,

$$\mathbf{I}^* = g(\mathbf{I}) \cdot \mathbf{I}. \tag{28}$$

Then, similar to [36], we normalize $\mathbf{I}^*$ and $\mathbf{T}$ to compute the final logits.

$$\mathbf{I} = \frac{\mathbf{I}^*}{|I^*|}, \mathbf{T} = \frac{\mathbf{T}}{|\mathbf{I}|}, \tag{29}$$

$$\hat{\mathbf{I}} = s \times \mathbf{I} \times \mathbf{T}^T, \hat{\mathbf{T}} = \hat{\mathbf{I}}^T. \tag{30}$$

where s is a scale parameter.

Now, we can utilize the ground truth, a vector $\tilde{\mathbf{y}} = [0, 1, 2, 3, \cdots, B]$

$$\begin{aligned} \ell_{cls}^I &= \ell(\hat{\mathbf{I}}, \tilde{\mathbf{y}}), \\ \ell_{cls}^T &= \ell(\hat{\mathbf{T}}, \tilde{\mathbf{y}}), \end{aligned} \tag{31}$$

where ℓ is CrossEntropy loss [57] while B is the number of images in a batch.

We only exchange parameters of adapters, w^g, and therefore in the server, we replace Eq. 26 with Eq. 32.

$$w^{g,*} = \sum_{i=1}^{N} \frac{n_i}{\sum_{j=1}^{N} n_j} w_i^g. \tag{32}$$

Since w^g contains substantially less amount of trainable parameters than w, FedCLIP saves computational costs and communication costs.

3.4 Summary

For clarity, we give a detailed description of FedCLIP in Algorithm 1. In Line 1, directly obtaining generalized and diversified features with fixed CLIP make it possible to utilize more prior knowledge of pretrained models. In Line 2, with adapters, we can concentrate on valuable information and eliminate the influence of redundant information in specific tasks. Rich prior knowledge and targeted attention make the ultimately extracted features more robust, effective, and adaptable, resulting in our method having good generalization and personalization capabilities. From Line 2 to Line 5, performing computation and transmission merely with adapters can save a lot of resources and ensure the efficiency of our method.

3.5 Discussion

Adapter is a common technique in transfer learning [18]. It is at a small scale and has a plug-and-play implementation. In this paper, we mainly focus on adaptations to image encoders. Actually, we also can add adapters to text encoders. We can even change the inputs of text encoders to incorporate more semantic information.

4 Experiments

In this section, we extensively evaluate FedCLIP in three common visual image classification benchmarks.

Algorithm 1 FedCLIP

Input: N clients' datasets $\{\mathcal{D}_i\}_{i=1}^{N}$, a pretained CLIP model consist of an image encoder, f^I, and a text encoder, f^T

Output: An adapter g

1: For client i, computer the corresponding features $I_i = f^I(\mathbf{X}_i), T_i = f^T(\mathbf{Y}_i)$
2: For client i, train the local adapter, g_i, according to Eq.5 to Eq.9
3: Send the current adapter g_i to the server
4: Aggregate adapters' parameters via Eq. 32 and obtain w^{g*}
5: Transmit w^{g*} to each client
6: Repeat steps $2 \sim 5$ until convergence

4.1 Datasets

PACS PACS [23] is a popular object classification benchmark. It is composed of four sub-datasets, including photo, art-painting, cartoon, and sketch. There exist $9,991$ images in total and the dataset contains 7 classes, including dog, elephant, giraffe, guitar, horse, house, and person. Large discrepancies in image styles widely exist among different sub-datasets. In this paper, we view each sub-dataset as a client. We choose three sub-datasets as participated clients while the rest served as the target client to evaluate generalization ability. For each participated client, we split the corresponding sub-dataset into three parts, 60% for training, 20% for validation, and the rest 20% for testing. Validation parts of data are used for model selection.

VLCS VLCS [10] is another widely accepted public image classification benchmark. It also consists of four sub-datasets (VOC2007, LabelMe, Caltech10, and SUN09). It contains $10,729$ instances with 5 classes. Feature shifts exist generally among different sub-datasets. Similar to PACS, four sub-datasets correspond to four clients. Three sub-datasets play the roles of participants while the rest one act as an upcoming client.

Office-Home Office-Home [48] is a larger image classification benchmark, which contains 65 classes. Office-Home comprises four sub-datasets (Art, Clipart, Product, and Real_World) with about $15,500$ images. The feature shifts from Office-Home mainly come from image styles and viewpoints, but they are much smaller than PACS. We assess methods on Office-Home in a similar manner to PACS.

4.2 Implementation Details and Comparison Methods

For these three common image classification benchmarks, we use the CLIP pre-trained model with ViT-B/32 [9] as the image encoder. For model training, we utilize cross-entropy loss and Adam optimizer. The learning rate is tuned from 5×10^{-5} to 5×10^{-3}. We set local update epochs as $E = 1$ where E means the number of training epochs in one round while we set the total communication round number as $R = 200$. Since, at each time, we set one sub-dataset as the target, i.e. upcoming client, there exist four tasks for each benchmark. We run three trials to record the average results. To better illustrate the function and necessity of using larger pretrained models, we also utilize a related small architecture, AlexNet [21], to perform some base federated learning methods.

We compare our method with two methods including a common federated learning method, FedAVG, and a method designed for non-iid data, FedProx.

1. FedAVG [33]. The server aggregates all client models' parameters. FedAVG will aggregate networks with several layers for AlexNet while FedAVG will aggregate both image encoders and text encoders for CLIP.

2. FedProx [25]. It adds a proximal term to FedAVG and allows the existence of slight differences between clients and the server.

Table 10: Generalization accuracy. **Bold** means the best.

Dataset		PACS					Office-Home						
Backbone	Method	A	C	P	S	AVG	Backbone	Method	A	C	P	R	AVG
AlexNet	FedAVG	31.54	43.69	44.55	36.29	39.02	AlexNet	FedAVG	15.70	17.00	31.56	28.99	23.31
	FedProx	29.79	46.80	44.67	35.12	39.09		FedProx	16.48	17.66	29.83	27.98	22.99
CLIP	FedAVG	53.08	80.08	90.00	76.99	75.04	CLIP	FedAVG	65.60	57.64	71.64	75.42	67.57
	FedProx	66.06	87.33	91.68	78.42	80.87		FedProx	65.60	57.64	71.64	75.42	67.57
	Ours	**96.34**	**97.91**	**99.76**	**85.59**	**94.90**		Ours	**78.00**	**63.69**	**87.52**	**87.79**	**79.25**

Table 11: Personalization accuracy. **Bold** means the best.

Dataset			PACS					Office-Home					
Target	BackBone	Method	C	P	S	AVG	Target	BackBone	Method	C	P	R	AVG
A	AlexNet	FedAVG	72.86	61.08	78.22	70.72	A	AlexNet	FedAVG	50.74	63.47	38.81	51.01
		FedProx	71.37	56.89	81.53	69.93			FedProx	51.78	66.74	40.07	52.86
	CLIP	FedAVG	76.28	86.83	42.42	68.51		CLIP	FedAVG	64.38	79.14	78.76	74.09
		FedProx	90.81	90.42	63.95	81.73			FedProx	64.38	79.14	78.76	74.09
		Ours	**97.65**	**99.40**	**86.75**	**94.60**			Ours	**68.61**	**87.37**	**88.06**	**81.35**
			A	P	S	AVG				A	P	R	AVG
C	AlexNet	FedAVG	46.45	66.17	75.67	62.76	C	AlexNet	FedAVG	23.51	61.78	41.56	42.28
		FedProx	47.19	64.07	77.45	62.90			FedProx	24.54	64.04	40.18	42.92
	CLIP	FedAVG	84.11	92.81	81.02	85.98		CLIP	FedAVG	73.81	80.38	80.48	78.23
		FedProx	86.06	92.81	85.61	88.16			FedProx	73.81	80.38	80.48	78.23
		Ours	**96.33**	**99.10**	**86.88**	**94.10**			Ours	**78.97**	**87.60**	**87.60**	**84.72**
			A	C	S	AVG				A	C	R	AVG
P	AlexNet	FedAVG	37.65	75.00	81.53	64.73	R	AlexNet	FedAVG	23.30	49.94	40.87	38.04
		FedProx	35.45	73.93	83.57	64.32			FedProx	21.03	48.91	39.84	36.59
	CLIP	FedAVG	83.13	93.38	84.97	87.16		CLIP	FedAVG	70.93	**68.73**	77.73	72.46
		FedProx	83.86	93.59	88.54	88.66			FedProx	70.93	**68.73**	77.73	72.46
		Ours	**97.56**	**97.65**	**86.75**	**93.99**			Ours	**78.35**	68.38	**87.94**	**78.23**
			A	C	P	AVG				A	C	P	AVG
S	AlexNet	FedAVG	53.30	68.80	66.17	62.76	P	AlexNet	FedAVG	22.27	49.14	58.51	43.31
		FedProx	52.32	69.66	66.47	62.82			FedProx	20.21	50.06	58.29	42.85
	CLIP	FedAVG	90.71	94.02	94.91	93.21		CLIP	FedAVG	69.07	66.21	77.79	71.02
		FedProx	91.44	94.66	95.81	93.97			FedProx	69.07	66.21	77.79	71.02
		Ours	**97.31**	**97.65**	**99.40**	**98.12**			Ours	**78.56**	**68.50**	**87.37**	**78.14**

4.3 Results

Generalization Ability We first evaluate the generalization ability of each method via accuracy on clients that do not participate in training. Table 10 shows the generalization results for each task on PACS and Office-Home. We have the following observations from these results. 1) Our method achieves the best generalization ability on average with remarkable improvements (about 14% for PACS and about 12% for Office-Home). Moreover, our method achieves the best generalization ability in each task, which demonstrates the excellent generalization ability of our method. 2) Compared to methods with AlexNet as the backbone, methods with CLIP as the backbone can obtain better performance. It demonstrates that large well-trained models can be able to bring better generalization. 3) Compared to methods with CLIP as the backbone, our method has a further improvement, which demonstrates that our method leverages prior knowledge better.

Table 12: Comprehensive average accuracy. **Bold** means the best

Datasets	PACS					Office-Home					
Backbone	AlexNet		CLIP			Backbone	AlexNet		CLIP		
Methods	FedAVG	FedProx	FedAVG	FedProx	Ours	Methods	FedAVG	FedProx	FedAVG	FedProx	Ours
A	60.93	59.89	64.65	77.81	**95.04**	A	42.18	43.77	71.97	71.97	**80.51**
C	57.99	58.88	84.50	87.95	**95.06**	C	35.96	36.60	73.08	73.08	**79.46**
P	59.68	59.41	87.87	89.42	**95.43**	P	36.42	34.90	72.26	72.26	**80.55**
S	56.14	55.89	89.16	90.08	**94.99**	R	39.73	39.13	72.12	72.12	**80.55**
AVG	58.69	58.52	81.55	86.32	**95.13**	AVG	38.57	38.60	72.36	72.36	**80.27**

Table 13: Comprehensive average accuracy on VLCS. **Bold** means the best

Backbone	AlexNet		CLIP		
Methods	FedAVG	FedProx	FedAVG	FedProx	Ours
C	62.13	61.37	72.48	68.57	**83.68**
L	63.01	63.77	75.04	76.50	**82.62**
S	63.15	63.59	68.13	75.50	**82.82**
V	62.32	62.04	69.55	70.09	**83.30**
AVG	62.65	62.69	71.30	72.67	**83.11**

Personalization Ability Then, we evaluate the personalization ability of each method via the accuracy on test data of each participating client. Table 11 shows the personalization results for each task on PACS and Office-Home. We also have some insightful observations. 1)Although all clients share the same adapter in our method, our method still achieves the best average accuracy. Moreover, FedCLIP almost achieves the best performance on each client for every task. 2) Compared to methods with AlexNet, corresponding methods with CLIP perform better overall. For CLIP-based methods, results are quite sensitive to hyperparameters, e.g. learning rate. And FedAVG has disappointing results on some specific clients. 3) Our method has the most use of prior knowledge since it achieves the stablest results.

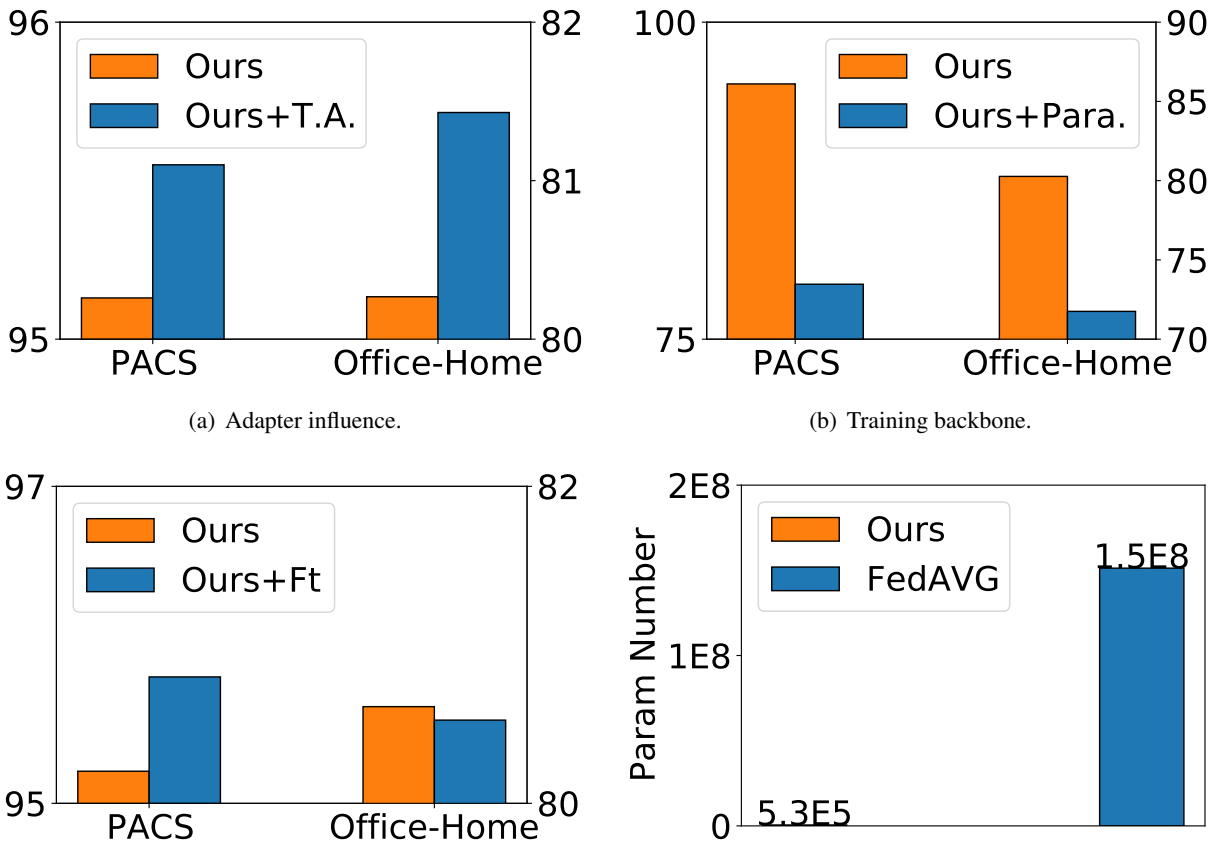
Comprehensive Ability Finally, taking into account the performance of both personalization and generalization, we provide an overall performance in Table 12. Without a doubt, our method achieves the best overall performance with significant improvements (about 9% for PACS and 8% for Office-Home). Compared to methods based on AlexNet, corresponding methods based on CLIP perform better.

More results on VLCS Due to space limitations, we only report comprehensive ability on VLCS. As shown in Table 13, our method still achieves the best performance with improvements of over 10%. Moreover, our method achieves the best in each task. The results prove the superiority of our method again.

4.4 Analysis

Can more adapters bring better performance? In our method, we only add one adapter to the image encoder. We can add another adapter to the text encoder. As shown in Figure 3(a), adding more adapters brings slight improvements. However, the improvements are so small that we need to assess whether it is necessary to do so since more adapters regularly mean more computational costs and more communication costs.

Can more trainable parameters bring better performance? If we train both adapters and the backbones, the results could be worse. Since CLIP models have a wealth of good information, it is not suitable to change



(a) Adapter influence.

(b) Training backbone.

(c) Finetune influence.

(d) Parameter counts.

Figure 3: Analysis on PACS.

parameters with only a few data for a specific task. Changes in CLIP with few data can destroy the feature extraction capabilities. As shown in Figure 3(b), we train more parameters but achieve worse performance.

Will finetuning bring better personalization? According to [54], finetuning can be a useful technique for better personalization. We also add experiments with finetune. As shown in Figure 3(c), finetune has no advance in personalization, which demonstrates that our method can be remarkable and robust when meeting non-iid.

Resource Cost Comparison The number of trainable parameters represents how many resources we need to cost in federated learning. As shown in Figure 3(d), our method merely has $5.3E5$ parameters while FedAVG with CLIP requires $1.5E8$ trainable parameters. Common methods via training whole networks have 283 times as many parameters as ours, which illustrates that our method is fast and resource-efficient.

5 Conclusion and Future Work

In this article, we propose FedCLIP, a fast generalization and personalization learning method for CLIP in federated learning. FedCLIP designs an attention based adapter to replace updating the whole model. Therefore, FedCLIP makes the most use of prior knowledge and saves computational costs and communication costs. Comprehensive experiments have demonstrated the superiority of FedCLIP. In the future, we plan to embed

FedCLIP into more architectures and design more flexible adapters for different tasks. We also plan to apply FedCLIP for heterogeneous architectures and more realistic applications.

References

[1] M. Z. Alom, T. M. Taha, C. Yakopcic, S. Westberg, P. Sidike, M. S. Nasrin, B. C. Van Esesn, A. A. S. Awwal, and V. K. Asari. The history began from alexnet: A comprehensive survey on deep learning approaches. arXiv preprint arXiv:1803.01164, 2018.

[2] M. Arjovsky, L. Bottou, I. Gulrajani, and D. Lopez-Paz. Invariant risk minimization. stat, 1050:27, 2020.

[3] S. Banabilah, M. Aloqaily, E. Alsayed, N. Malik, and Y. Jararweh. Federated learning review: Fundamentals, enabling technologies, and future applications. Information processing & management, 59(6):103061, 2022.

[4] R. Bommasani, D. A. Hudson, E. Adeli, R. Altman, S. Arora, S. von Arx, M. S. Bernstein, J. Bohg, A. Bosselut, E. Brunskill, et al. On the opportunities and risks of foundation models. arXiv preprint arXiv:2108.07258, 2021.

[5] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. Advances in neural information processing systems, 33:1877–1901, 2020.

[6] D. Caldarola, B. Caputo, and M. Ciccone. Improving generalization in federated learning by seeking flat minima. In Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXIII, pages 654–672. Springer, 2022.

[7] H.-Y. Chen and W.-L. Chao. On bridging generic and personalized federated learning for image classification. In International Conference on Learning Representations, 2022.

[8] Y. Chen, W. Lu, X. Qin, J. Wang, and X. Xie. Metafed: Federated learning among federations with cyclic knowledge distillation for personalized healthcare. arXiv preprint arXiv:2206.08516, 2022.

[9] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In International Conference on Learning Representations, 2021.

[10] C. Fang, Y. Xu, and D. N. Rockmore. Unbiased metric learning: On the utilization of multiple datasets and web images for softening bias. In Proceedings of the IEEE International Conference on Computer Vision, pages 1657–1664, 2013.

[11] P. Foret, A. Kleiner, H. Mobahi, and B. Neyshabur. Sharpness-aware minimization for efficiently improving generalization. In International Conference on Learning Representations, 2021.

[12] L. Gao, H. Fu, L. Li, Y. Chen, M. Xu, and C.-Z. Xu. Feddc: Federated learning with non-iid data via local drift decoupling and correction. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 10112–10121, 2022.

[13] M. W. Gardner and S. Dorling. Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. Atmospheric environment, 32(14-15):2627–2636, 1998.

[14] T. Guo, S. Guo, J. Wang, and W. Xu. Promptfl: Let federated participants cooperatively learn prompts instead of models–federated learning in age of foundation model. arXiv preprint arXiv:2208.11625, 2022.

[15] S. Gupta, K. Ahuja, M. Havaei, N. Chatterjee, and Y. Bengio. Fl games: A federated learning framework for distribution shifts. In Workshop on Federated Learning: Recent Advances and New Challenges (in Conjunction with NeurIPS 2022), 2022.

[16] K. Han, Y. Wang, H. Chen, X. Chen, J. Guo, Z. Liu, Y. Tang, A. Xiao, C. Xu, Y. Xu, et al. A survey on vision transformer. IEEE transactions on pattern analysis and machine intelligence, 45(1):87–110, 2022.

[17] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 770–778, 2016.

[18] W. Hou, H. Zhu, Y. Wang, J. Wang, T. Qin, R. Xu, and T. Shinozaki. Exploiting adapters for cross-lingual low-resource speech recognition. IEEE ACM Trans. Audio Speech Lang. Process., 30:317–329, 2022.

[19] Z. Huang, H. Wang, E. P. Xing, and D. Huang. Self-challenging improves cross-domain generalization. In Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16, pages 124–140. Springer, 2020.

[20] N. Inkster. China's cyber power. Routledge, 2018.

[21] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In NeurIPS, volume 25, pages 1097–1105, 2012.

[22] S. Lee, H. Park, D. U. Kim, J. Kim, M. Boboev, and S. Baek. Image-free domain generalization via clip for 3d hand pose estimation. In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, pages 2934–2944, 2023.

[23] D. Li, Y. Yang, Y.-Z. Song, and T. M. Hospedales. Deeper, broader and artier domain generalization. In Proceedings of the IEEE international conference on computer vision, pages 5542–5550, 2017.

[24] L. Li, Y. Fan, M. Tse, and K.-Y. Lin. A review of applications in federated learning. Computers & Industrial Engineering, 149:106854, 2020.

[25] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith. Federated optimization in heterogeneous networks. Proceedings of Machine Learning and Systems, 2:429–450, 2020.

[26] X. Li, M. JIANG, X. Zhang, M. Kamp, and Q. Dou. Fedbn: Federated learning on non-iid features via local batch normalization. In International Conference on Learning Representations, 2021.

[27] J. Liu, J. Huang, Y. Zhou, X. Li, S. Ji, H. Xiong, and D. Dou. From distributed machine learning to federated learning: A survey. Knowledge and Information Systems, 64(4):885–917, 2022.

[28] W. Liu, Y. Wen, B. Raj, R. Singh, and A. Weller. Sphereface revived: Unifying hyperspherical face recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, 45(2):2458–2474, 2022.

[29] W. Lu, Y. Chen, J. Wang, and X. Qin. Cross-domain activity recognition via substructural optimal transport. Neurocomputing, 454:65–75, 2021.

[30] W. Lu, J. Wang, and Y. Chen. Local and global alignments for generalizable sensor-based human activity recognition. In IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2022.

[31] W. Lu, J. Wang, Y. Chen, S. Pan, C. Hu, and X. Qin. Semantic-discriminative mixup for generalizable sensor-based cross-domain activity recognition. IMWUT, 2022.

[32] W. Lu, J. Wang, Y. Chen, X. Qin, R. Xu, D. Dimitriadis, and T. Qin. Personalized federated learning with adaptive batchnorm for healthcare. IEEE Transactions on Big Data, 2022.

[33] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas. Communication-efficient learning of deep networks from decentralized data. In Artificial Intelligence and Statistics, pages 1273–1282. PMLR, 2017.

[34] N. Paluru, A. Dayal, H. B. Jenssen, T. Sakinis, L. R. Cenkeramaddi, J. Prakash, and P. K. Yalavarthy. Anam-net: Anamorphic depth embedding-based lightweight cnn for segmentation of anomalies in covid-19 chest ct images. IEEE Transactions on Neural Networks and Learning Systems, 32(3):932–946, 2021.

[35] Z. Qu, X. Li, R. Duan, Y. Liu, B. Tang, and Z. Lu. Generalized federated learning via sharpness aware minimization. In International Conference on Machine Learning, pages 18250–18280. PMLR, 2022.

[36] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. In International conference on machine learning, pages 8748–8763. PMLR, 2021.

[37] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, et al. Improving language understanding by generative pre-training. 2018.

[38] A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, and I. Sutskever. Zero-shot text-to-image generation. In International Conference on Machine Learning, pages 8821–8831. PMLR, 2021.

[39] N. Rieke, J. Hancox, W. Li, F. Milletari, H. R. Roth, S. Albarqouni, S. Bakas, M. N. Galtier, B. A. Landman, K. Maier-Hein, et al. The future of digital health with federated learning. NPJ digital medicine, 3(1):1–7, 2020.

[40] N. Rodríguez-Barroso, D. Jiménez-López, M. V. Luzón, F. Herrera, and E. Martínez-Cámara. Survey on federated learning threats: Concepts, taxonomy on attacks and defences, experimental study and challenges. Information Fusion, 90:148–173, 2023.

[41] A. G. Roy, S. Siddiqui, S. Pölsterl, N. Navab, and C. Wachinger. Braintorrent: A peer-to-peer environment for decentralized federated learning. arXiv, 2019.

[42] I. H. Sarker. Machine learning: Algorithms, real-world applications and research directions. SN computer science, 2(3):160, 2021.

[43] F. Sattler, S. Wiedemann, K.-R. Müller, and W. Samek. Robust and communication-efficient federated learning from non-iid data. IEEE transactions on neural networks and learning systems, 31(9):3400–3413, 2019.

[44] C. Sun, X. Qiu, Y. Xu, and X. Huang. How to fine-tune bert for text classification? In Chinese Computational Linguistics: 18th China National Conference, CCL 2019, Kunming, China, October 18–20, 2019, Proceedings 18, pages 194–206. Springer, 2019.

[45] I. Tenison, S. A. Sreeramadas, V. Mugunthan, E. Oyallon, E. Belilovsky, and I. Rish. Gradient masked averaging for federated learning. arXiv preprint arXiv:2201.11986, 2022.

[46] I. Tenney, D. Das, and E. Pavlick. Bert rediscovers the classical nlp pipeline. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 4593–4601, 2019.

[47] E. A. van Dis, J. Bollen, W. Zuidema, R. van Rooij, and C. L. Bockting. Chatgpt: five priorities for research. Nature, 614(7947):224–226, 2023.

[48] H. Venkateswara, J. Eusebio, S. Chakraborty, and S. Panchanathan. Deep hashing network for unsupervised domain adaptation. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 5018–5027, 2017.

[49] P. Voigt and A. Von dem Bussche. The eu general data protection regulation (gdpr). A Practical Guide, 1st Ed., Cham: Springer International Publishing, 10:3152676, 2017.

[50] J. Wang, Y. Chen, S. Hao, X. Peng, and L. Hu. Deep learning for sensor-based activity recognition: A survey. Pattern Recognition Letters, 119:3–11, 2019.

[51] J. Wang, C. Lan, C. Liu, Y. Ouyang, T. Qin, W. Lu, Y. Chen, W. Zeng, and P. Yu. Generalizing to unseen domains: A survey on domain generalization. IEEE Transactions on Knowledge and Data Engineering, 2022.

[52] S. Warnat-Herresthal, H. Schultze, K. L. Shastry, S. Manamohan, S. Mukherjee, V. Garg, R. Sarveswara, K. Händler, P. Pickkers, N. A. Aziz, et al. Swarm learning for decentralized and confidential clinical machine learning. Nature, 594(7862):265–270, 2021.

[53] Q. Yang, Y. Liu, T. Chen, and Y. Tong. Federated machine learning: Concept and applications. ACM Transactions on Intelligent Systems and Technology (TIST), 10(2):1–19, 2019.

[54] T. Yu, E. Bagdasaryan, and V. Shmatikov. Salvaging federated learning by local adaptation. arXiv preprint arXiv:2002.04758, 2020.

[55] H. Yuan, W. R. Morningstar, L. Ning, and K. Singhal. What do we mean by generalization in federated learning? In The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022. OpenReview.net, 2022.

[56] L. Yuan, Y. Chen, T. Wang, W. Yu, Y. Shi, Z.-H. Jiang, F. E. Tay, J. Feng, and S. Yan. Tokens-to-token vit: Training vision transformers from scratch on imagenet. In Proceedings of the IEEE/CVF international conference on computer vision, pages 558–567, 2021.

[57] Z. Zhang and M. Sabuncu. Generalized cross entropy loss for training deep neural networks with noisy labels. Advances in neural information processing systems, 31, 2018.

Reconciling Security and Communication Efficiency in Federated Learning

Karthik Prasad*† Sayan Ghosh*† Graham Cormode†
Ilya Mironov† Ashkan Yousefpour† Pierre Stock†
†Meta AI

Abstract

Cross-device Federated Learning is an increasingly popular machine learning setting to train a model by leveraging a large population of client devices with high privacy and security guarantees. However, communication efficiency remains a major bottleneck when scaling federated learning to production environments, particularly due to bandwidth constraints during uplink communication. In this paper, we formalize and address the problem of compressing client-to-server model updates under the Secure Aggregation primitive, a core component of Federated Learning pipelines that allows the server to aggregate the client updates without accessing them individually. In particular, we adapt standard scalar quantization and pruning methods to Secure Aggregation and propose Secure Indexing, a variant of Secure Aggregation that supports quantization for extreme compression. We establish state-of-the-art results on LEAF benchmarks in a secure Federated Learning setup with up to 40× compression in uplink communication with no meaningful loss in utility compared to uncompressed baselines.

1 Introduction

Federated Learning (FL) is a distributed machine learning (ML) paradigm that trains a model across a number of participating entities holding local data samples. In this work, we focus on cross-device FL that harnesses a large number (up to hundreds of millions) of edge devices with disparate characteristics such as availability, compute, memory, or connectivity resources [31]. Two challenges to the success of cross-device FL are privacy and scalability. FL was originally motivated for improving privacy since data points remain on client devices. However, as with other forms of ML, information about training data can be extracted via membership inference or reconstruction attacks on a trained model [11, 12], or leaked through local updates [40, 19]. Consequently, Secure Aggregation (SECAGG) protocols were introduced to prevent the server from directly observing individual client updates, which is a major vector for information leakage [8, 27]. Additional mitigations such as Differential Privacy (DP) may be required to offer further protection against attacks [17, 1], as discussed in Section 6.

Ensuring scalability to populations of heterogeneous clients is the second challenge for FL. Indeed, wall-clock training times are highly correlated with increasing model and batch sizes [27], even with recent efforts such as FedBuff [41], and communication overhead between the server and clients dominates model convergence time.

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

*Equal contribution. Correspondence to pstock@fb.com.

Consequently, compression techniques were used to reduce the communication bandwidth while maintaining model accuracy. However, a fundamental problem has been largely overlooked in the literature: in their native form, standard compression methods such as scalar quantization and pruning are not compatible with SECAGG. This makes it challenging to ensure both security and communication efficiency.

We address this gap by adapting compression techniques to make them compatible with SECAGG. We focus on compressing uplink updates from clients to the server for three reasons. First, uplink communication is more sensitive and so is subject to a high security bar, whereas downlink updates broadcast by the server are deemed public. Second, upload bandwidth is generally more restricted than download bandwidth. For instance, according to a recent FCC report, the ratio of download to upload speeds for DSL and cable providers[1] in the US ranges between 3× to 20× [18]. Efficient uplink communication brings several benefits beyond speeding up convergence: lowering communication cost reduces selection bias due to under-sampling clients with limited connectivity, improving fairness and inclusiveness. It shrinks the carbon footprint of FL, the fraction of which attributable to communication can reach 95% [45]. In summary, we present the following contributions:

- We highlight the fundamental mismatch between two critical components of the FL stack: SECAGG protocols and uplink compression mechanisms.
- We formulate solutions by imposing a linearity constraint on the decompression operator, as illustrated in Figure 1 in the case of TEE-based SECAGG.
- We adapt the popular scalar quantization and (random) pruning compression methods for compatibility with the FL stack that require no changes to the SECAGG protocol.
- For extreme uplink compression without compromising security, we propose Secure Indexing (SECIND), a variant of SECAGG that supports product quantization.

2 Related Work

Communication is identified as a primary efficiency bottleneck in FL, especially in the cross-device FL setting [31]. This has led to significant interest in reducing FL's communication requirements. In what follows, we refer to a local model update in distributed training as a gradient, including updates from multiple local training steps.

Efficient Distributed Optimization. There is a large body of literature on reducing the communication cost for distributed training. [49] proposes quantizing gradients to one bit while carrying the quantization error forward across mini-batches with error feedback. Similarly, [57] proposes layer-wise ternary gradients and [6] suggests using only the sign of the gradients. Gradient sparsity is another related area that is extensively studied [56, 2, 36, 46, 42]. For instance, [14] and [22] explore adapting the degree of sparsity to the distribution of local client data. Another method, QSGD, tunes the quantization level to trade possibly higher variance gradients for reduced communication bandwidth [3]. Researchers also studied structured and sketched model updates [32]. For example, [54] proposes expressing gradients as a linear combination of basis vectors common to all workers and [55] propose to cluster the gradients and to implement error correction on the client side. Besides gradient compression, other methods such as [52, 26] reduce the communication cost by partitioning the model such that each client learns a portion of it, while [24] proposes training small models and periodically distilling them to a larger central model. However, as detailed in Section 3 and below, most of the proposed methods are not readily compatible with SECAGG and cannot be used in secure FL.

Bi-directional Compression. In addition to uplink gradient compression, a line of work also focuses on downlink model compression. In a non-distributed setup, [61, 16] demonstrates that it is possible to meaningfully train with low bit-width models and gradients. In FL, [30] proposes adapting the model size to the device to reduce

[1] FL is typically restricted to using unmetered connections, usually over Wi-Fi [27].

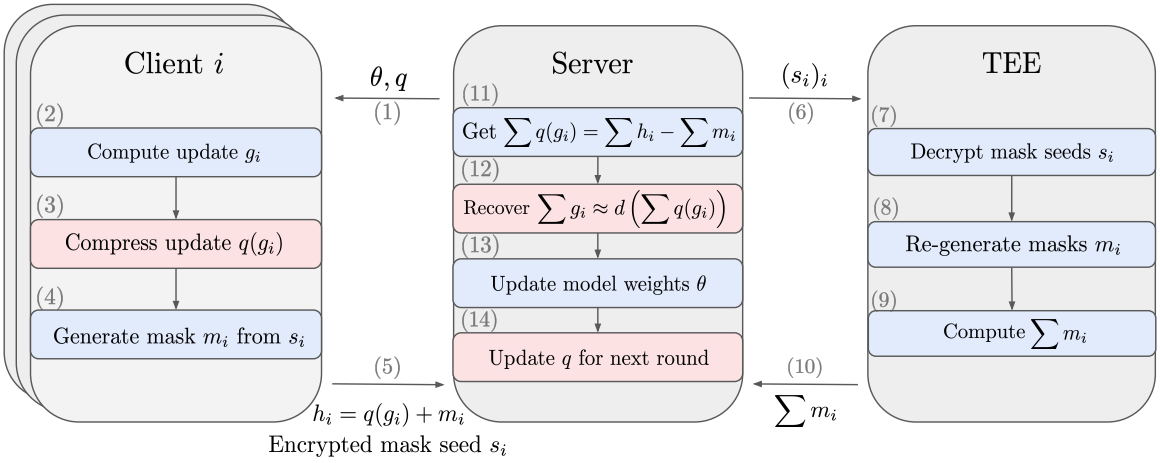


Figure 1: Summary of the proposed approach for one FL round, where we omit the round dependency and Differential Privacy (DP) for clarity. Blue boxes denote standard steps and red boxes denote additional steps for uplink compression. Client i computes local model update g_i, compresses it with the compression operator q, and encrypts it by adding a random mask m_i in the compressed domain, hence reducing the uplink bandwidth (steps 2–4). The server recovers the aggregate in the compressed domain by leveraging any SECAGG protocol (steps 7–13, with a TEE-based SECAGG, see Section 3.1). Since the decompression operator d is linear, the server can convert the aggregate back to the non-compressed domain, up to compression error (step 12). As with the model weights θ, the compression operator q are also periodically updated and broadcast by the server (step 14). In Section 4, we apply the proposed method to scalar quantization and pruning without impacting SECAGG and propose Secure Indexing, a variant of SECAGG for extreme uplink compression with product quantization. See Section 3.1 for details about SECAGG and Section 6 for a discussion on DP.

both communication and computation overhead. Since the local models are perturbed due to compression, researchers propose adapting the optimization algorithm for better convergence [37, 48, 51, 60, 4, 43]. Finally, pre-conditioning models during FL training can allow for quantized on-device inference, as demonstrated for non-distributed training by [21, 33]. As stated in Section 1, we do not focus on downlink model compression since uplink bandwidth is the main communication bottleneck and since SECAGG only involves uplink communication.

Aggregation in the Compressed Domain. In the distributed setting, [59] propose to leverage both gradient compression and parallel aggregation by performing the ring all-reduce operation in the compressed domain and decompressing the aggregate. To do so, the authors exploit temporal correlations of the gradients to design a linear compression operator. Another method, PowerSGD [53], leverages a fast low-rank gradient compressor. However, both aforementioned methods are not evaluated in the FL setup and do not mention SECAGG. Indeed, the proposed methods focus on decentralized communication between the workers by leveraging the all-reduce operation. Moreover, PowerSGD uses (stateful) error feedback on all distributed nodes, which is not readily adaptable to cross-device FL when clients generally participate in a few (not necessarily consecutive) rounds. Finally, [47] proposes FetchSGD, a compression method using sketching, which is compatible with SECAGG.

3 Background

In this section, we recall the SECAGG protocol first, then the compression methods that we wish to adapt to SECAGG, namely, scalar quantization, pruning, and product quantization.

3.1 Secure Aggregation

SECAGG refers to a class of protocols that allow the server to aggregate client updates without accessing them individually. While SECAGG alone does not entirely prevent client data leakage, it is a powerful and widely-used component of current at-scale cross-device FL implementations [31]. Two main approaches exist in practice: software-based protocols relying on Multiparty Computation (MPC) [8, 5, 58], and those that leverage hardware implementations of Trusted Execution Environments (TEEs) [27].

SECAGG relies on additive masking, where clients protect their model updates g_i by adding a uniform random mask m_i to it, guaranteeing that each client's masked update is statistically indistinguishable from any other value. At aggregation time, the protocol ensures that all the masks are canceled out. For instance, in an MPC-based SECAGG, the pairwise masks cancel out within the aggregation itself, since for every pair of users i and j, after they agree on a matched pair of input perturbations, the masks $m_{i,j}$ and $m_{j,i}$ are constructed so that $m_{i,j} = -m_{j,i}$. Similarly and as illustrated in Fig. 1, in a TEE-based SECAGG, the server receives $h_i = g_i + m_i$ from each client as well as the sum of the masks $\sum_i m_i$ from the TEE and recovers the sum of the updates as $\sum_i g_i = \sum_i h_i - \sum_i m_i$. We defer the discussion of DP noise addition by SECAGG protocols to Section 6.

Finite Group. SECAGG requires that the plaintexts—client model updates—be elements of a finite group, while the inputs are real-valued vectors represented with floating-point types. This requirement is usually addressed by converting client updates to fixed-point integers and operating in a finite domain (modulo 2^p) where p is typically set in prior literature to 32 bits. The choice of SECAGG bit-width p must balance communication costs with the accuracy loss due to rounding and overflows.

Minimal Complexity. TEE-based protocols offer greater flexibility in how individual client updates can be processed; however, the code executed inside TEE is part of the trusted computing base (TCB) for all clients. In particular, it means that this code must be stable, auditable, defects- and side-channel-free, which severely limits its complexity. Hence, in practice, we prefer compression techniques that are either oblivious to SECAGG's implementation or require minimal changes to the TCB.

3.2 Compression Methods

In this subsection, we consider a matrix $W \in \mathbb{R}^{C_{\text{in}} \times C_{\text{out}}}$ representing the weights of a linear layer to discuss three major compression methods with distinct compression/accuracy tradeoffs and identify the challenges SECAGG faces to be readily amenable to these popular quantization algorithms.

3.2.1 Scalar Quantization

Uniform scalar quantization maps floating-point weight w to 2^b evenly spaced bins, where b is the number of bits. Given a floating-point scale $s > 0$ and an integer shift parameter z called the zero-point, we map any floating-point parameter w to its nearest bin indexed by $\{0, \ldots, 2^b - 1\}$:

$$w \mapsto \text{clamp}(\text{round}(w/s) + z, [0, 2^b - 1]).$$

The tuple (s, z) is often referred to as the quantization parameters (`qparams`). With $b = 8$, we recover the popular `int8` quantization scheme [28], while setting $b = 1$ yields the extreme case of binarization [16]. The quantization parameters s and z are usually calibrated after training a model with floating-point weights using the minimum and maximum values of each layer. The compressed representation of weights W consists of the `qparams` and the integer representation matrix W_q where each entry is stored in b bits. Decompressing any integer entry w_q of W_q back to floating point is performed by applying the (linear) operator $w_q \mapsto s \times (w_q - z)$.

Challenge. The discrete domain of quantized values and the finite group required by SECAGG are not natively compatible because of the overflows that may occur at aggregation time. For instance, consider the extreme case of binary quantization, where each value is replaced by a bit. We can represent these bits in SECAGG with $p = 1$, but the aggregation will inevitably result in overflows.

3.2.2 Pruning

Pruning is a class of methods that remove parts of a model such as connections or neurons according to some pruning criterion, such as weight magnitude ([34, 23]; see [7] for a survey). [32] demonstrate client update compression with random sparsity for federated learning. Motivated by previous work and the fact that random masks do not leak information about the data on client devices, we will leverage random pruning of client updates in the remainder of this paper. A standard method to store a sparse matrix is the coordinate list (COO) format[2], where only the non-zero entries are stored (in floating point or lower precision), along with their integer coordinates in the matrix. This format is compact, but only for a large enough compression ratio, as we store additional values for each non-zero entry. Decompression is performed by re-instantiating the uncompressed matrix with both sparse and non-sparse entries.

Challenge. Pruning model updates on the client side is an effective compression approach as investigated in previous work. However, the underlying assumption is that clients have different masks, either due to their seeds or dependency on client update parameters (*e.g.,* weight magnitudes). This is a challenge for SECAGG as aggregation assumes a dense compressed tensor, which is not possible to construct when the coordinates of non-zero entries are not the same for all clients.

3.2.3 Product Quantization

Product quantization (PQ) is a compression technique developed for nearest-neighbor search [29] that can be applied for model compression [50]. Here, we show how we can re-formulate PQ to represent model updates. We focus on linear layers and refer the reader to [50] for adaptation to convolutions. Let the block size be d (say, 8), the number of codewords be k (say, 256) and assume that the number of input channels, C_{in}, is a multiple of d. To compress W with PQ, we evenly split its columns into subvectors or blocks of size $d \times 1$ and learn a codebook via k-means to select the k codewords used to represent the $C_{\text{in}} \times C_{\text{out}}/d$ blocks of W. PQ with block size $d = 1$ amounts to non-uniform scalar quantization with $\log_2 k$ bits per weight.

The PQ-compressed matrix W is represented with the tuple (C, A), where C is the codebook of size $k \times d$ and A gives the assignments of size $C_{\text{in}} \times C_{\text{out}}/d$. Assignments are integers in $[0, k-1]$ and denote which codebook a subvector was assigned to. To decompress the matrix (up to reshaping), we index the codebook with the assignments, written in PyTorch-like notation as $\widehat{W} = C[A]$.

Challenge. There are several obstacles to making PQ compatible with SECAGG. First, each client may have a different codebook, and direct access to these codebooks is needed to decode each client's message. Even if all clients share a (public) codebook, the operation to take assignments to produce an (aggregated) update is not linear, and so cannot be directly wrapped inside SECAGG.

4 Method

In this section, we propose solutions to reconcile security (SECAGG) and communication efficiency. Our approach is to modify compression techniques to share some hyperparameters globally across all clients so that aggregation can be done by uniformly combining each client's response, while still ensuring that there is scope to achieve accurate compressed representations. As detailed below, each of the proposed methods offers the same level of security as standard SECAGG without compression.

4.1 Secure Aggregation and Compression

We propose to compress the uplink model updates through a compression operator q, whose parameters are round-dependent but the same for all clients participating in the same round. Then, we will add a random mask

[2]See the torch.sparse documentation, `https://pytorch.org/docs/stable/sparse.html`.

m_i to each quantized client update $q(g_i)$ in the compressed domain, thus effectively reducing uplink bandwidth while ensuring that $h_i = q(g_i) + m_i$ is statistically indistinguishable from any other representable value in the finite group (see Section 3.1). In this setting, SECAGG allows the server to recover the aggregate of the client model updates in the compressed domain: $\sum_i q(g_i)$. If the decompression operator d is linear, the server is able to recover the aggregate in the non-compressed domain, up to quantization error, as illustrated in Figure 1:

$$d\left(\sum_i h_i - \sum_i m_i\right) = d\left(\sum_i q(g_i)\right) = \sum_i d(q(g_i)) \approx \sum_i g_i.$$

The server periodically updates the quantization and decompression operator parameters, either from the aggregated model update, which is deemed public, or by emulating a client update on some similarly distributed public data. Once these parameters are updated, the server broadcasts them to the clients for the next round. This adds overhead to the downlink communication payload, however, this is negligible compared to the downlink model size to transmit. For instance, for scalar quantization, q is entirely characterized by one `fp32` scale and one `int32` zero-point per layer, the latter of which is unnecessary in the case of a symmetric quantization scheme. Finally, this approach is compatible with both synchronous FL methods such as FedAvg [39] and asynchronous methods such as FedBuff [41] as long as SECAGG maintains the mapping between the successive versions of quantization parameters and the corresponding client updates.

4.2 Application

Next, we show how we adapt scalar quantization and random pruning with no changes required to SECAGG. We illustrate our point with TEE-based SECAGG while these adapted uplink compression mechanisms are agnostic of the SECAGG mechanism. Finally, we show how to obtain extreme uplink compression by proposing a variant of SECAGG, which we call SECIND. This variant supports product quantization and is provably secure.

4.2.1 Scalar Quantization and Secure Aggregation

As detailed in Section 3.2.1, a model update matrix g_i compressed with scalar quantization is given by an integer representation in the range $[0, 2^b - 1]$ and by the quantization parameters scale (s) and zero-point (z). A sufficient condition for the decompression operator to be linear is to broadcast common quantization parameters per layer for each client. Denote $q(g_i)$ as the integer representation of quantized client model update g_i corresponding to a particular layer for client $1 \leq i \leq N$. Set the scale of the decompression operator to s and its zero-point to z/N. Then, the server is able to decompress as follows (where the decompression operator is defined in Section 3.2.1):

$$d\left(\sum_i q(g_i)\right) = s\sum_i q(g_i) - \tfrac{z}{N} = \sum_i \left(s(q(g_i)) - z\right) \approx \sum_i g_i$$

Recall that all operations are performed in a finite group. Therefore, to avoid overflows at aggregation time, we quantize with a bit-width b but take SECAGG bit-width $p > b$, thus creating a margin for potential overflows (see Section 5.3). This approach is related to the fixed-point aggregation described in [8, 27], but we calibrate the quantization parameters and perform the calibration per layer and periodically, unlike the related approaches.

Privacy, Security and Bandwidth. Scales and zero points are determined from public data on the server. Downlink overhead is negligible: the server broadcasts the per-layer quantization parameters. The upload bandwidth is p bits per weight, where p is the SECAGG finite group size (Section 3.1). Since the masks m_i are chosen in the integer range $[0, 2^p - 1]$, any masked integer representation taken modulo 2^p is statistically indistinguishable from any other vector.

4.2.2 Pruning and Secure Aggregation

To enable linear decompression with random pruning, all clients will share a common pruning mask for each round. This can be communicated compactly before each round as a seed for a pseudo-random function. This

Algorithm 2 Secure Indexing (SECIND)

procedure SECUREINDEXING(C) ▷ This happens inside the TEE
  Receive common codebook C from server ▷ C is periodically updated by the server
  Initialize histograms $H_{m,n}$ to 0 ▷ Each histogram for block (m, n) has size k
  for each client i **do**
    Receive and decrypt assignment matrix A^i
    for each block index (m, n) **do**
      $r \leftarrow A^i_{m,n}$ ▷ Recover assignment of client i for block (m, m)
      $H_{m,n}[r] \leftarrow H_{m,n}[r] + 1$ ▷ Update global count for codeword index r
  Send back histograms $H_{m,n}$ to the server

pruning mask seed is different from the SECAGG mask seed introduced in Section 3.1 and has a distinct role. Each client uses the pruning seed to reconstruct a pruning mask, prunes their model update g_i, and only needs to encrypt and transmit the unpruned parameters. The trade-off here is that some parameters are completely unobserved in a given round, as opposed to traditional pruning. SECAGG operates as usual and the server receives the sum of the tensor of unpruned parameters computed by participating clients in the round, which it can expand using the mask seed. We denote the pruning operator as ϕ applied to the original model update g_i, and the decompression operator as d applied to a compressed tensor $\phi(g_i)$. Decompression is an expansion operation equivalent to multiplication with a sparse permutation matrix P_i whose entries are dependent on the i'th client's mask seed. Crucially, when all clients share the same mask seed within each round, we have $P_i = P$ for all i and linearity of decompression is maintained:

$$d\left(\sum_i \phi(g_i)\right) = P\left(\sum_i \phi(g_i)\right) = \sum_i P_i \phi(g_i) = \sum_i d(\phi(g_i)) \approx \sum_i g_i.$$

Privacy, Security and Bandwidth. Since the mask is random, no information leaks from the pruning mask. The downlink overhead (the server broadcasts one integer mask seed) is negligible. The upload bandwidth is simply the size of the sparse client model updates. Finally, there is no loss in security since each client uses standard SECAGG mechanism on the non-pruned entries.

4.2.3 Product Quantization and Secure Indexing

We next describe the Secure Indexing (SECIND) primitive, and discuss how to instantiate it. Recall that with PQ, each layer has its own codebook C as explained in Section 4. Let us fix one particular layer compressed with codebook C, containing k codewords. We assume that C is common to all clients participating in the round. Consider the assignment matrix of a given layer $(A^i)_{m,n}$ for client i. From these, we seek to build the assignment histograms $H_{m,n} \in \mathbb{R}^k$ that satisfy $H_{m,n}[r] = \sum_i \mathbf{1}\left(A^i_{m,n} = r\right)$, where the indicator function $\mathbf{1}$ satisfies $\mathbf{1}\left(A^i_{m,n} = r\right) = 1$ if $A^i_{m,n} = r$ and 0 otherwise. A Secure Indexing primitive will produce $H_{m,n}$ while ensuring that no other information about client assignments or partial aggregations is revealed. The server receives assignment histograms from SECIND and is able to recover the aggregated update for each block indexed by (m, n) as $\sum_r H_{m,n}[r] \cdot C[r]$. We describe how SECIND can be implemented with a TEE in Algorithm 2. Each client encrypts the assignment matrix, for instance with additive masking as described in Section 3.1, and sends it to the TEE via the server. Hence, the server does not have access to the plaintexts client-specific assignments. TEE decrypts each assignment matrix and for each block indexed by (m, n) produces the assignment histogram. Compared to SECAGG, where the TEE receives an encrypted seed per client (a few bytes per client) and sends back the sum of the masks m_i (same size as the considered model), SECIND receives the (masked) assignment matrices and sends back histograms for each round. SECIND can be implemented in other models, offering different trust paradigms, such as the multi-party computation setting (using two or more servers to operate on

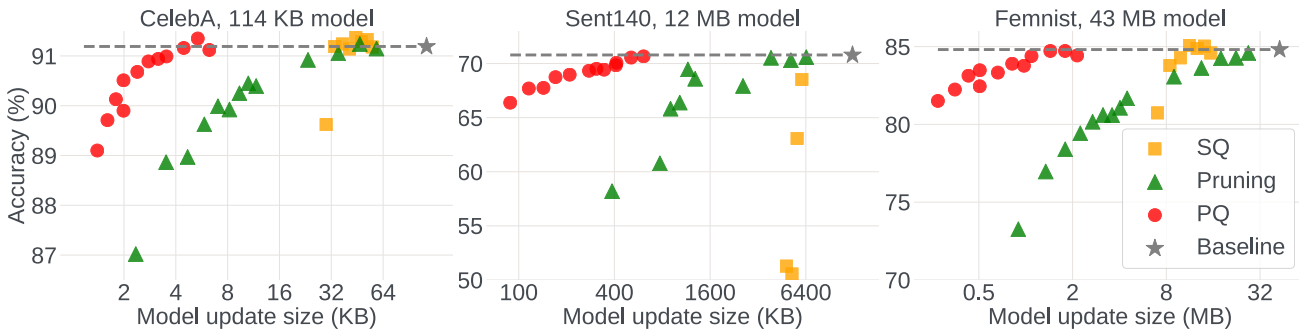


Figure 2: We adapt scalar quantization (SQ) and pruning to the SECAGG protocol to enable efficient and secure uplink communications. We also present results for product quantization (PQ) under the proposed novel SECIND protocol. The x axis is log-scale and represents the uplink message size. Baseline refers to SECAGG FL run without any uplink compression, shown as a horizontal line for easier comparison. Model size is indicated in plot titles. Uncompressed client updates are as large as the models when $p = 32$ (see Sec 3.1, shown as stars).

shares of the input). Encoding inputs as shares of one-hot vectors would lose the advantages of compression. Instead, each client can send evaluations of distributed point functions to encode each assignment [9]. These are represented compactly, but may require longer codewords to overcome the overheads.

Privacy, Security and Bandwidth. Codebooks are computed from public data while individual assignments are never revealed to the server. The downlink overhead of sending the codebooks is negligible as demonstrated in Section 5. The upload bandwidth in the TEE implementation is the assignment size, represented in k bits (the number of codewords). For instance, with a block size $d = 8$ and $k = 32$ codewords, assignment storage costs are 5 bits per 8 weights, which converts to 0.625 bits per weight. The tradeoff compared to non-secure PQ is the restriction to a global codebook for all clients (instead of one tailored to each client), and the need to instantiate SECIND instead of SECAGG. Since the assignments are encrypted before being sent to the TEE, there is no loss in security. Here, any encryption mechanism (not necessarily relying on additive masking) would work.

5 Experiments

We evaluate the performance of the proposed approaches when adapted to SECAGG protocols. We study the relationship between uplink compression and model accuracy for the LEAF benchmark tasks. For scalar and product quantization we also analyze the impact of refresh rate for compression parameters on model performance.

5.1 Experimental Setup

We follow the setup of [41] and use the FLSim library for our experiments . All experiments are run on a single V100 GPU 16 GB (except for Sent140 where we use one V100 32 GB) and typically take a few hours to run.

Tasks. We run experiments on three datasets from the LEAF benchmark [10]: CelebA [38], Sent140 [20] and FEMNIST [35]. For CelebA, we train the same convolutional classifier as [41] with BatchNorm layers replaced by GroupNorm layers and 9,343 clients. For Sent140, we train an LSTM classifier for binary sentiment analysis with $59,400$ clients. For FEMNIST, we train a GroupNorm version of the ResNet18 [25] for digit classification with 3,550 clients. We do not compress biases and norm layers due to their small overhead.

Baselines. We focus here on the (synchronous) FedAvg approach although, as explained in Section 4, the proposed compression methods can be readily adapted to asynchronous FL aggregation protocols. As in prior work, we keep the number of clients per round to at most 100, a small fraction of the total considered population

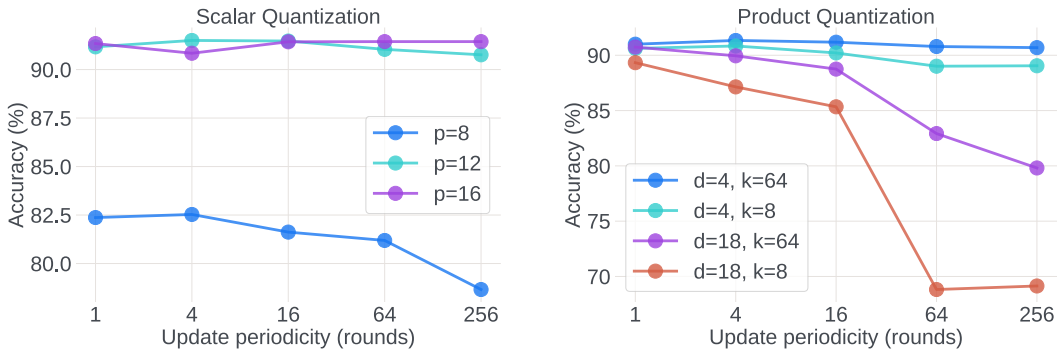


Figure 3: Impact of the refresh rate of the compression operator by the server on the CelebA dataset. **Left**: scalar quantization (quantization parameters), fixing the quantization bit-width $b = 8$ (p denotes the SECAGG bit-width). **Right**: for product quantization (codebooks), where k denotes the number of codewords and d the block size.

size [15, 13]. We report the average and standard deviation of accuracy over three independent runs for all tasks at different uplink byte sizes corresponding to various configurations of the compression operator.

Implementation Details. We refer the reader to [44] for full experiment details. The downlink overhead of sending the per-layer codebooks for product quantization is negligible. The convergence time in terms of rounds is similar for PQ runs and the non-compressed baseline/ Note that outside a simulated environment, the wall-clock time convergence for PQ runs would be lower than the baseline since uplink communication would be more efficient, hence faster.

5.2 Results and Comparison with Prior Work

Results for efficient and secure uplink communications are displayed in Figure 2. We observe that PQ yields a consistently better trade-off curve between model update size and accuracy. For instance, on CelebA, PQ achieves $\times 30$ compression with respect to the non-compressed baseline at iso-accuracy. The iso-accuracy compression rate is $\times 32$ on Sent140 and $\times 40$ on FEMNIST (see [44] for detailed tables). Scalar quantization accuracy degrades significantly for larger compression rates due to the overflows at aggregation. Pruning gives intermediate tradeoffs between scalar quantization and product quantization.

The line of work that develops FL compression techniques is exemplified by FetchSGD [47] as detailed in Section 2, where the authors do not address SECAGG. Their results are not directly comparable to ours due to incomparable experimental setups (e.g., datasets and architectures). However, Figure 6 [44] mentions upload compression rates at iso-accuracy that are weaker than those obtained here with product quantization.

5.3 Ablation Studies

We investigate the influence of the frequency of updates of the compression operator q for scalar quantization and pruning, and study the influence of the SECAGG bit-width p on the number of overflows for scalar quantization.

Update frequency of the compression operators. In Figure 3, we show that for scalar quantization, the update periodicity only plays a role with low SECAGG bit-width values p compared to the quantization bit-width b. For product quantization, the update periodicity plays an important role for aggressive compression setups corresponding to large block sizes d or to a smaller number of codewords k. For pruning, we measure the impact of masks that are refreshed periodically. We observed that if we refresh the compression operator more frequently, staleness is reduced, leading to accuracy improvements.

Overflows for scalar quantization. As discussed in Section 4.2.1, we choose the SECAGG bit-width p to be greater than quantization bit-width b in order to avoid aggregation overflows. While it suffices to set p to be $\lceil \log_2 n_c \rceil$ more than b, where n_c is the number of clients participating in the round, reducing p is desirable to

reduce uplink size. We studied the impact of p on the percentage of parameters that suffer overflows, and observed that there is a benefit to having some non-zero overflow margin size, but no clear correlation between margin size and accuracy.

6 Concluding Remarks

In this paper, we reconcile efficiency and security for uplink communication in Federated Learning. We propose to adapt existing compression mechanisms such as scalar quantization and pruning to the secure aggregation protocol by imposing a linearity constraint on the decompression operator. Our experiments demonstrate that we can adapt both quantization and pruning mechanisms to obtain a high degree of uplink compression with minimal degradation in performance and higher security guarantees. For achieving the highest rates of compression, we introduce SECIND, a variant of SECAGG well-suited for TEE-based implementation that supports product quantization while maintaining a high security bar. As mentioned in Section 1, we may want both SECAGG and Differential Privacy [1] to realize the full promise of FL as a privacy-enhancing technology. While our primary focus is on enabling efficient and secure uplink communication, we emphasize that the proposed approaches are compatible with user-level DP. For instance, at the cost of increasing the complexity of the trusted computing base, DP noise can be added natively by the TEE with our modified random pruning or scalar quantization approaches. For PQ and SECIND, we can have the TEE to add noise in the assignment space (*i.e.,* outputting a noisy histogram), or to map the histogram to the codeword space and add noise there. Each option offers a different tradeoff between privacy, trust, and accuracy; we leave detailed evaluation to future work.

References

[1] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang. Deep learning with differential privacy. In ACM CCS, 2016.

[2] A. F. Aji and K. Heafield. Sparse communication for distributed gradient descent. In EMNLP, 2017.

[3] D. Alistarh, D. Grubic, J. Li, R. Tomioka, and M. Vojnovic. QSGD: Communication-efficient SGD via gradient quantization and encoding. In NeurIPS, 2017.

[4] M. M. Amiri, D. Gunduz, S. R. Kulkarni, and H. V. Poor. Federated learning with quantized global model updates. CoRR, 2006.10672, 2020.

[5] J. H. Bell, K. A. Bonawitz, A. Gascón, T. Lepoint, and M. Raykova. Secure single-server aggregation with (poly) logarithmic overhead. In ACM CCS, 2020.

[6] J. Bernstein, Y.-X. Wang, K. Azizzadenesheli, and A. Anandkumar. signSGD: Compressed optimisation for non-convex problems. In ICML, PMLR, 2018.

[7] D. Blalock, J. J. Gonzalez Ortiz, J. Frankle, and J. Guttag. What is the state of neural network pruning? In MLSys, 2020.

[8] K. A. Bonawitz, F. Salehi, J. Konečný, B. McMahan, and M. Gruteser. Federated learning with autotuned communication-efficient secure aggregation. In ACSCC. IEEE, 2019.

[9] E. Boyle, N. Gilboa, and Y. Ishai. Function secret sharing: Improvements and extensions. In ACM CCS, 2016.

[10] S. Caldas, P. Wu, T. Li, J. Konečný, H. B. McMahan, V. Smith, and A. Talwalkar. LEAF: A benchmark for federated settings. CoRR, abs/1812.01097, 2018.

[11] N. Carlini, S. Chien, M. Nasr, S. Song, A. Terzis, and F. Tramèr. Membership inference attacks from first principles. CoRR, abs/2112.03570, 2021.

[12] N. Carlini, F. Tramèr, E. Wallace, M. Jagielski, A. Herbert-Voss, K. Lee, A. Roberts, T. B. Brown, D. Song, Ú. Erlingsson, A. Oprea, and C. Raffel. Extracting training data from large language models. In USENIX Security Symposium, 2021.

[13] Z. Charles, Z. Garrett, Z. Huo, S. Shmulyian, and V. Smith. On large-cohort training for federated learning. CoRR, 2106.07820, 2021.

[14] C. Chen, J. Choi, D. Brand, A. Agrawal, W. Zhang, and K. Gopalakrishnan. AdaComp: Adaptive residual gradient compression for data-parallel distributed training. In AAAI, 2018.

[15] M. Chen, R. Mathews, T. Ouyang, and F. Beaufays. Federated learning of out-of-vocabulary words. CoRR, 1903.10635, 2019.

[16] M. Courbariaux, Y. Bengio, and J.-P. David. BinaryConnect: Training deep neural networks with binary weights during propagations. CoRR, 1511.00363, 2015.

[17] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In Proceedings of the Third Conference on Theory of Cryptography, 2006.

[18] FCC. The eleventh Measuring Broadband America fixed broadband report, 2021.

[19] J. Geiping, H. Bauermeister, H. Dröge, and M. Moeller. Inverting gradients—How easy is it to break privacy in federated learning? In NeurIPS, 2020.

[20] A. Go, R. Bhayani, and L. Huang. Twitter sentiment classification using distant supervision. CS224N Project Report, Stanford, 2009.

[21] S. Gupta, A. Agrawal, K. Gopalakrishnan, and P. Narayanan. Deep learning with limited numerical precision. In ICML, 2015.

[22] P. Han, S. Wang, and K. K. Leung. Adaptive gradient sparsification for efficient federated learning: An online learning approach. In ICDCS, 2020.

[23] B. Hassibi and D. G. Stork. Second order derivatives for network pruning: Optimal Brain Surgeon. In NeurIPS, 1992.

[24] C. He, M. Annavaram, and S. Avestimehr. Group knowledge transfer: Federated learning of large CNNs at the edge. In NeurIPS, 2020.

[25] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In IEEE CVPR, 2016.

[26] C. Hu, W. Bao, D. Wang, and F. Liu. Dynamic adaptive DNN surgery for inference acceleration on the edge. IEEE INFOCOM, 2019.

[27] D. Huba, J. Nguyen, K. Malik, R. Zhu, M. Rabbat, A. Yousefpour, C.-J. Wu, H. Zhan, P. Ustinov, H. Srinivas, K. Wang, A. Shoumikhin, J. Min, and M. Malek. Papaya: Practical, private, and scalable federated learning. In MLSys, 2022.

[28] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In IEEE CVPR, June 2018.

[29] H. Jégou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. IEEE Trans. Pattern Anal. Mach. Intell., 33(1):117–128, 2011.

[30] Y. Jiang, S. Wang, B. J. Ko, W. Lee, and L. Tassiulas. Model pruning enables efficient federated learning on edge devices. CoRR, abs/1909.12326, 2019.

[31] P. Kairouz et al. Advances and open problems in federated learning. Found. Trends Mach. Learn., 14(1–2), 2021.

[32] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtarik, A. T. Suresh, and D. Bacon. Federated learning: Strategies for improving communication efficiency. In NIPS Workshop on Private Multi-Party Machine Learning, 2016.

[33] R. Krishnamoorthi. Quantizing deep convolutional networks for efficient inference. CoRR, 1806.08342, 2018.

[34] Y. Le Cun, J. S. Denker, and S. A. Solla. Optimal brain damage. In NeurIPS, 1989.

[35] Y. LeCun and C. Cortes. MNIST handwritten digit database. http://yann.lecun.com/exdb/mnist/, 2010.

[36] Y. Lin, S. Han, H. Mao, Y. Wang, and W. Dally. Deep gradient compression: Reducing the communication bandwidth for distributed training. In ICLR, 2018.

[37] X. Liu, Y. Li, J. Tang, and M. Yan. A double residual compression algorithm for efficient distributed learning. In AISTATS, 2020.

[38] Z. Liu, P. Luo, X. Wang, and X. Tang. Deep learning face attributes in the wild. In IEEE ICCV, 2015.

[39] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas. Communication-efficient learning of deep networks from decentralized data. In AISTATS, 2017.

[40] L. Melis, C. Song, E. D. Cristofaro, and V. Shmatikov. Exploiting unintended feature leakage in collaborative learning. In IEEE S&P, 2019.

[41] J. Nguyen, K. Malik, H. Zhan, A. Yousefpour, M. Rabbat, M. Malek, and D. Huba. Federated learning with buffered asynchronous aggregation. In AISTATS, 2022.

[42] T. Parcollet, J. Fernandez-Marques, P. PB Gusmao, Y. Gao, and N. D. Lane. ZeroFL: Efficient on-device training for federated learning with local sparsity. In ICLR, 2022.

[43] C. Philippenko and A. Dieuleveut. Preserved central model for faster bidirectional compression in distributed settings.

In NeurIPS, 2021.

[44] K. Prasad, S. Ghosh, G. Cormode, I. Mironov, A. Yousefpour, and P. Stock. Reconciling security and communication efficiency in federated learning. CoRR, 2207.12779, 2022.

[45] X. Qiu, T. Parcollet, J. Fernandez-Marques, P. P. B. de Gusmao, D. J. Beutel, T. Topal, A. Mathur, and N. D. Lane. A first look into the carbon footprint of federated learning. arXiv, 2102.07627, 2021.

[46] C. Renggli, S. Ashkboos, M. Aghagolzadeh, D. Alistarh, and T. Hoefler. SparCML: High-performance sparse communication for machine learning. In SC, 2019.

[47] D. Rothchild, A. Panda, E. Ullah, N. Ivkin, I. Stoica, V. Braverman, J. Gonzalez, and R. Arora. FetchSGD: Communication-efficient federated learning with sketching. In ICML, 2020.

[48] F. Sattler, S. Wiedemann, K.-R. Müller, and W. Samek. Robust and communication-efficient federated learning from non-i.i.d. data. IEEE Transactions on Neural Networks and Learning Systems, 31:3400–3413, 2020.

[49] F. Seide, H. Fu, J. Droppo, G. Li, and D. Yu. 1-bit stochastic gradient descent and its application to data-parallel distributed training of speech DNNs. In INTERSPEECH, 2014.

[50] P. Stock, A. Joulin, R. Gribonval, B. Graham, and H. Jégou. And the bit goes down: Revisiting the quantization of neural networks. In ICLR, 2020.

[51] H. Tang, C. Yu, X. Lian, T. Zhang, and J. Liu. DOUBLESQUEEZE: Parallel stochastic gradient descent with double-pass error-compensated compression. In ICML, 2019.

[52] P. Vepakomma, O. Gupta, T. Swedish, and R. Raskar. Split learning for health: Distributed deep learning without sharing raw patient data, 2018.

[53] T. Vogels, S. P. Karimireddy, and M. Jaggi. PowerSGD: Practical low-rank gradient compression for distributed optimization. In NeurIPS, 2019.

[54] H. Wang, S. Sievert, S. Liu, Z. Charles, D. Papailiopoulos, and S. Wright. ATOMO: Communication-efficient learning via atomic sparsification. In NeurIPS, 2018.

[55] J. Wang, H. Qi, A. S. Rawat, S. Reddi, S. Waghmare, F. X. Yu, and G. Joshi. Fedlite: A scalable approach for federated learning on resource-constrained clients. CoRR, 2201.11865, 2022.

[56] J. Wangni, J. Wang, J. Liu, and T. Zhang. Gradient sparsification for communication-efficient distributed optimization. In NeurIPS, 2018.

[57] W. Wen, C. Xu, F. Yan, C. Wu, Y. Wang, Y. Chen, and H. Li. TernGrad: Ternary gradients to reduce communication in distributed deep learning. In NeurIPS, 2017.

[58] C. Yang, J. So, C. He, S. Li, Q. Yu, and S. Avestimehr. LightSecAgg: Rethinking secure aggregation in federated learning. In MLSys, 2022.

[59] M. Yu, Z. Lin, K. Narra, S. Li, Y. Li, N. S. Kim, A. Schwing, M. Annavaram, and S. Avestimehr. GradiVeQ: Vector quantization for bandwidth-efficient gradient aggregation in distributed CNN training. In NeurIPS, 2018.

[60] S. Zheng, Z. Huang, and J. T. Kwok. Communication-efficient distributed blockwise momentum SGD with error-feedback. In NeurIPS, 2019.

[61] S. Zhou, Z. Ni, X. Zhou, H. Wen, Y. Wu, and Y. Zou. DoReFa-Net: Training low bitwidth convolutional neural networks with low bitwidth gradients. CoRR, abs/1606.06160, 2016.

Accelerated Federated Optimization with Quantization

Yeojoon Youn[†] Bhuvesh Kumar[†] Jacob Abernethy[†]
[†] Georgia Institute of Technology {yjyoun92,bhuvesh,prof}@gatech.edu

Abstract

Federated optimization is a new form of distributed training on very large datasets that leverages many devices each containing local data. While decentralized computation can lead to significant speed-ups due to parallelization, some centralization is still required: devices must aggregate their parameter updates through synchronization across the network. The potential for communication bottleneck is significant. The two main methods to tackle this issue are (a) smarter optimization that decreases the frequency of communication rounds and (b) using compression techniques such as quantization and sparsification to reduce the number of bits machines need to transmit. In this paper, we provide a novel algorithm, ***Fed****erated optimization algorithm with* ***A****cceleration and* ***Q****uantization (FedAQ), with improved theoretical guarantees by combining an accelerated method of federated averaging, reducing the number of training and synchronization steps, with an efficient quantization scheme that significantly reduces communication complexity. We show that in a homogeneous strongly convex setting, FedAQ achieves a linear speedup in the number of workers M with only $\tilde{\mathcal{O}}(M^{\frac{1}{3}})$ communication rounds, significantly smaller than what is required by other quantization-based federated optimization algorithms. Moreover, we empirically verify that our algorithm performs better than current methods.*

1 Introduction

Federated learning (FL) has attracted much attention from both academia and industry due to the increasing demand for large-scale distributed machine learning systems and preserving privacy-sensitive data on local devices such as smartphones and IoT devices. In federated learning, a number of clients collaboratively learn the global objective function by communicating with a central server without sharing any locally stored data in each local device. The research in Federated learning has identified four major challenges: communication efficiency, systems heterogeneity, statistical heterogeneity, and privacy [19]. In this paper, we focus on communication efficiency that is of primary interest in cross-device settings when there is a heavy communication burden with many edge computing devices and limited network bandwidth. Two of the most widely used methods to reduce the communication cost are federated averaging optimization and randomized compression techniques.

In federated averaging (FedAvg) [26], also called *local SGD*, each client locally updates its model with multiple stochastic gradient descent (SGD) steps, and a server aggregates model updates of clients. The server updates its own model parameters by averaging client models and then broadcasts the server parameters to all clients. This enables FL systems to achieve high communication efficiency with infrequent synchronization while showing better performance than distributed large mini-batch SGD [25]. Due to the significant empirical success

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

Table 14: Summary of Results on the Convergence Rate and Communication Required for Linear Speedup. M is the number of devices, T is the number of total parallel iterations, and K is the number of communication rounds, q is a quantization parameter (Assumption 1), d_{quant} is the number of bits used to quantize, d_{full} is the number of bits required when there is no quantization ($d_{\text{full}} \gg d_{\text{quant}}$). [43] and FedAQ send two iterates per communication round as other algorithms to achieve acceleration (See line 11 in Algorithm 3), we multiply d_{full} and d_{quant} by 2 for bits communicated for a linear speedup. The presented results of [9] are newly obtained (section 5.5).

Algorithm	Convergence rate	Communication rounds for $\tilde{\mathcal{O}}(\frac{1}{T})$ convergence with linear speedup	Bits communicated for linear speedup
FedPAQ [27]	$\mathcal{O}(\frac{1+q}{K} + \frac{T}{K^2})$	Not possible	Not possible
FedCOMGATE [9]	$\tilde{\mathcal{O}}(\frac{1+q}{MT} + \frac{1}{TK})$	$\tilde{\mathcal{O}}(\frac{M}{1+q})$	$\tilde{\mathcal{O}}(\frac{M}{1+q}) \cdot d_{\text{quant}}$
FedAC [43]	$\tilde{\mathcal{O}}(\frac{1}{MT} + \frac{1}{TK^3})$	$\tilde{\mathcal{O}}(M^{\frac{1}{3}})$	$\tilde{\mathcal{O}}(M^{\frac{1}{3}}) \cdot 2d_{\text{full}}$
FedAQ (Corollary 23)	$\tilde{\mathcal{O}}(\frac{1+q}{MT} + \frac{1+q}{TK^3})$	$\tilde{\mathcal{O}}(M^{\frac{1}{3}})$	$\tilde{\mathcal{O}}(M^{\frac{1}{3}}) \cdot 2d_{\text{quant}}$

of FedAvg, researchers have proposed an interesting theoretical question: To what extent can we minimize the number of synchronizations in order to both guarantee convergence and achieve linear speedup in the number of workers M[1]? For the strongly-convex and homogeneous settings, [15] was able to achieve a linear speedup in M with $\tilde{\mathcal{O}}(M)$ communication rounds, which is the state-of-the-art result for FedAvg convergence analysis. However, even with this progress on theoretical guarantees of FedAvg, it remains unclear whether further improvements on convergence time and communication efficiency can be achieved.

Applying acceleration methods to FL has led to improved convergence, with [43] providing a faster version of FedAvg with provably stronger bounds. For the strongly-convex and homogeneous setting, their algorithm achieves a linear speedup in M with only $\tilde{\mathcal{O}}(M^{\frac{1}{3}})$ communication rounds. Hence, the accelerated version of federated averaging requires a much smaller number of communication rounds than FedAvg to achieve the same accuracy. At present, this remains the best result for strongly-convex and homogeneous local data distribution settings. In addition to reducing the required number of communication rounds, another powerful way to build communication-efficient FL systems is to reduce the number of bits that need to be transmitted at each synchronization. [27, 9] have shown that such compression techniques, which include *quantization*, reduce communication costs and guarantee convergence (See Table 14).

In this work, we provide a novel algorithm, **Fed**erated optimization algorithm with **A**cceleration and **Q**uantization (FedAQ), to solve the severe communication bottleneck problem in FL systems. FedAQ is the first federated optimization algorithm that successfully incorporates multiple local update schemes, acceleration, and quantization for master-worker topology. Although these three key desiderata of Federal Learning systems have individually been shown to build communication-efficient FL systems, it is not obvious if or how acceleration techniques can lead to faster convergence even for quantization-based methods. We answer this question by showing that FedAQ converges for strongly-convex and homogeneous local data distribution settings without any additional strong assumptions.

Let T be the number of total parallel iterations, K be the number of total communication rounds. We compare our results to previous methods in Table 14, and highlight the following contributions:

1. FedAQ has a convergence rate of $\tilde{\mathcal{O}}(\frac{1+q}{MT} + \frac{1+q}{TK^3})$ which is better than the $\tilde{\mathcal{O}}(\frac{1+q}{MT} + \frac{1}{TK})$ convergence of [9], the state of the art in quantization based methods. Here q is a parameter that measures the effectiveness of the quantization scheme (see Assumption 1). This allows FedAQ to obtain linear speedup with only $\tilde{\mathcal{O}}(M^{\frac{1}{3}})$ communication rounds whereas [9] requires $\tilde{\mathcal{O}}(\frac{M}{1+q})$ rounds. The faster convergence in number of

[1]Linear speedup in the number of workers is a desirable property in parallel computing which implies that the task takes half as much time if the number of workers are doubled.

communication rounds also implies that FedAQ can achieve better convergence than [9] by using many fewer communication rounds. Thus, although FedAQ sends two iterates in each communication round, that is the bits communicated in each round are twice many compared to [9] for the same level of quantization, FedAQ requires much smaller total communication costs due to the large reduction in synchronization rounds.

2. When comparing FedAQ to Accelerated Federated learning, we observe that FedAQ has similar convergence and requires the same number of communication rounds as [43]. In each communication round of [43], every client sends the complete iterates to the server without any quantization. To effectively obtain a convergence rate of $\tilde{\mathcal{O}}(\frac{1}{MT})$, it needs to send each value with a precision of $\tilde{\mathcal{O}}(\frac{1}{MT})$, requiring $d_{\text{full}} = \mathcal{O}(\log(MT))$ bits. In comparison, if we use the low precision quantizer (See section 3 Example 1) given by [1], FedAQ needs to send only $d_{\text{quant}} = O(\log \frac{1}{q})$ bits [2] for each value. Since q is a constant, $d_{\text{quant}} \ll d_{\text{full}}$. The extra $1 + q$ term in the convergence for FedAQ can be offset by scaling the number of local updates by $1 + q$, which is cheaper than expensive data communication. Thus, FedAQ obtains the same convergence as [43] using as many communication rounds but by sending many fewer bits per round.

Finally, we empirically verify that our algorithm exhibits better performance than baselines, FedPAQ [27], FedCOMGATE [9], FedAC [43], and FedAvg [26] on classical vision datasets such as MNIST [18] and CIFAR-10 [17].

2 Related Works

The first guarantee for FedAvg, showing that it converges at the same rate as mini-batch SGD in strongly convex scenarios, was shown by [30] in the IID setting. The further convergence analysis of FedAvg for non-convex functions was laid out in a number of published works [36, 8, 42]. Followup work has managed to remove unnecessary assumptions, such as uniformly bounded gradients, to achieve better convergence rates [36, 31, 7, 15, 40]. Moreover, [20, 10, 21, 15, 14] define scenarios that depart from the IID framework, analyzing the convergence of FedAvg and its variants in settings with heterogeneous data distributions.

Reducing the transmitted bits between a server and clients through compression techniques is pivotal to saving communication costs in federated learning. This motivates researchers to develop various compression techniques such as sparsification and quantization without significantly sacrificing accuracy [16, 1, 32, 39, 4, 34, 33, 11, 3, 28]. [27] shows near-optimal theoretical guarantees of the first federated optimization algorithm that incorporates federated averaging, partial node participation, and quantization in homogeneous local data distribution settings. [9] further provide improved convergence rates for both homogeneous and heterogeneous settings.

We can achieve better communication efficiency by applying acceleration methods into client updates. [43] has proposed the first provable acceleration of FedAvg that achieves a linear speedup with the fewest communication rounds. Several other works aim to achieve communication efficiency by using momentum or adaptive optimizers [41, 13, 37]. It is important to note that our work is not the first to combine acceleration and quantization. [23, 24], for example, propose compressed and accelerated distributed optimization methods that are neither stochastic nor FedAvg variants. [29] proposes communication efficient momentum SGD for decentralized optimization. [22, 38] show that distributed and federated versions of adaptive optimizers along with gradient compression can lead to similar convergence rates as their non-compressed counterparts. But these works do not achieve the core result of the present paper, which is the reduced communication complexity via a faster convergence rate and a linear speedup with the small number of communication rounds. To the best of our knowledge, FedAQ is the first accelerated version of federated averaging for master-worker topology that successfully integrates a quantization scheme and provides rigorous convergence guarantees.

[2] More details on this are discussed in section 5.5.3

3 Problem Setup

In this paper, we build our algorithm based on federated learning with captain-worker topology where M local devices contain their own local data, and a server aggregates local parameter updates without sharing any data during synchronization rounds. Since we focus on homogeneous local data distribution settings for the convergence analysis of our algorithm, we define the distributed stochastic optimization problem as below.

$$\min_{w\in\mathbb{R}^d} F(w) := \mathbb{E}_{z\sim\mathcal{D}}[f(w;z)]$$

In our convergence analysis, we assume F is strongly-convex. Each client can access F at w via oracle $\nabla f(w;z)$ because all clients have the same loss function f. Also, every local device has the same local data distribution $\mathcal{D}$. Moreover, we use the full participation of nodes for local updates and synchronizations.

3.1 Assumptions

Let us clarify assumptions on the unbiased quantizer Q, the global objective function F, and the unbiased gradient estimator ∇f.

Assumption 1: The variance of the unbiased quantizer Q is bounded by the squared of l_2-norm of its argument, i.e., $\mathbb{E}[Q(x)|x] = x,\ \mathbb{E}[\|Q(x)-x\|^2|x] \le q\|x\|^2$.

For example, a well-known randomized quantizer which satisfies Assumption 1 is low-precision quantizer in [1].

Example 1. (Low-precision quantizer) Given $x\in\mathbb{R}^d$, the quantizer $Q:\mathbb{R}^d\to\mathbb{R}^d$ is defined by

$$Q_i(x) = \text{sign}(x_i)\cdot\|x\|\cdot\xi_i(x,s),\quad i\in[d]$$

ξ_i is defined as below.

$$\xi_i(x,s) = \begin{cases} \frac{l+1}{s}, & \text{with probability } \frac{|x_i|}{\|x\|}s - l \\ \frac{l}{s}, & \text{o/w} \end{cases}$$

s is the number of quantization levels. $l\in[0,s)$ is an integer which satisfies $\frac{|x_i|}{\|x\|}\in[\frac{l}{s},\frac{l+1}{s})$.

Assumption 2: F is μ-strongly convex, i.e., $F(w_1)\ge F(w_2)+\langle\nabla F(w_2), w_1-w_2\rangle+\frac{1}{2}\mu\|w_1-w_2\|^2$ for any $w_1,w_2\in\mathbb{R}^d$.

Assumption 3: F is L-smooth, i.e., $F(w_1)\le F(w_2)+\langle\nabla F(w_2), w_1-w_2\rangle+\frac{1}{2}L\|w_1-w_2\|^2$ for any $w_1,w_2\in\mathbb{R}^d$.

Assumption 4: $\nabla f(w;\xi)$ is unbiased and variance bounded, i.e., $\mathbb{E}_\xi[\nabla f(w;\xi)]=\nabla F(w)$, $\mathbb{E}_\xi[\|\nabla f(w;\xi)-\nabla F(w)\|^2]\le\sigma^2$ for any $w\in\mathbb{R}^d$.

3.2 Notation

We use τ, K to respectively denote the number of local updates and total communication rounds, which means the total number of iterations T at each node satisfies $T=K\tau$. Since we consider a strongly-convex case, we can find the optimal point w^* and denote the optimal function value as $F^*:=F(w^*)$. The local parameter $w_{k,t}^m$ indicates the parameter of the m-th local model after kth synchronization followed by t local SGD updates. There are other types of parameters such as $w_{k,t}^{\text{ag},m}$ and $w_{k,t}^{\text{md},m}$, and we obtain two types of parameters w_k and w_k^{ag} in the server side after kth synchronization. More details on these parameters will be discussed in the next section.

4 FedAQ Algorithm

We propose a novel communication efficient algorithm that combines an accelerated variant of federated averaging and an efficient quantization scheme. Our FedAQ algorithm has two main parts: (1) multiple accelerated local updates and (2) communication with quantization. Both components contribute to achieving better communication efficiency than other previous federated algorithms. The entire process is summarized in Algorithm 3.

4.1 Multiple Accelerated Local Updates

The FedAvg algorithm, proposed by [26], is widely used for federated learning to improve communication efficiency by reducing communication rounds with multiple local SGD updates. [43] provide FedAC that replaces the stochastic gradient updates of FedAvg by accelerated version of SGD by [6] resulting in a linear speedup in M with fewer communication rounds than FedAvg.

Thus, we apply the FedAC scheme to multiple updates of each local model. Since previous quantization-based federated optimization algorithms are FedAvg variants with no acceleration, the accelerated method enables our algorithm to gain better communication efficiency than others.

As you can see in Algorithm 3, we need two more local parameters $w_{k,t}^{\mathrm{ag},m}$ and $w_{k,t}^{\mathrm{md},m}$ for acceleration in addition to the main local parameter $w_{k,t}^m$. $w_{k,t}^{\mathrm{ag},m}$ aggregates the past iterates, and the gradients are queried at the auxiliary parameter $w_{k,t}^{\mathrm{md},m}$. While typical FL algorithms without acceleration only have a learning rate η as their hyperparameter, the general acceleration scheme makes our algorithm flexible due to four hyperparameters $\alpha, \beta, \eta, \gamma$. α, β are hyperparameters related to coupling coefficients, and η, γ stand for learning rates respectively for $w_{k,t}^{\mathrm{ag},m}, w_{k,t}^m$. The flexibility of hyperparameters enables the fast convergence speed of FedAQ, but naively chosen hyperparameters also cause unstable training of FedAQ. We discuss the exact choice of hyperparameters in section 5. Unlike FedAC, that requires each client to communicate the exact iterates to the server with high precision, we discuss in the following subsection how FedAQ incorporates quantization techniques to reduce communication cost.

4.2 Communication with Quantization

In cross-device federated learning, a large amount of communicated messages from a number of devices and the limited communication bandwidth can lead to severe communication bottlenecks. Therefore, in this scenario, an efficient quantization scheme can significantly reduce the size of communicated messages and make communication between local devices and a server faster. We apply the same unbiased quantizer used in [9] that satisfies Assumption 1.

In contrast with other quantization-based federated optimization algorithms [27, 9], the algorithmic novelty of FedAQ is based on applying quantization to two model parameter updates, which is required in order to simultaneously reduce the frequency of communication and the volume of communicated bits. To the best of our knowledge, this is the first quantization-based method that achieves the accelerated rate with the dramatic reduction in communication cost. To be specific on the communication process, after each client m obtains $w_{k,\tau}^m, w_{k,\tau}^{\mathrm{ag},m}$ through τ accelerated local iterations, each client quantizes the difference between $w_{k,\tau}^m, w_{k,\tau}^{\mathrm{ag},m}$ and the most recent server models w_k, w_k^{ag}. Then, a server aggregates $Q(w_{k,\tau}^m - w_k), Q(w_{k,\tau}^{\mathrm{ag},m} - w_k^{\mathrm{ag}})$ from all clients. After dequantizing those messages, the server obtains the following new models $w_{k+1}, w_{k+1}^{\mathrm{ag}}$ and broadcasts them back to each client.

Algorithm 3 Federated Accelerated SGD with Quantization (FedAQ)

1: **Input:** $\alpha, \beta, \eta, \gamma$, initial vector $w_0 = w_{0,0}^{\text{ag},m} = w_{0,0}^m$ for all devices $m \in [M]$
2: **for** $k = 0, \cdots, K-1$ **do**
3: **for** each client m in parallel **do**
4: $w_{k,0}^m \leftarrow w_k, w_{k,0}^{\text{ag},m} \leftarrow w_k^{\text{ag}}$
5: **for** $t = 0, \cdots, \tau - 1$ **do**
6: $w_{k,t}^{\text{md},m} \leftarrow \beta^{-1} w_{k,t}^m + (1 - \beta^{-1}) w_{k,t}^{\text{ag},m}$
7: $g_{k,t}^m \leftarrow \nabla f(w_{k,t}^{\text{md},m}, \xi_{k,t}^m)$
8: $w_{k,t+1}^{\text{ag},m} \leftarrow w_{k,t}^{\text{md},m} - \eta g_{k,t}^m$
9: $w_{k,t+1}^m \leftarrow (1 - \alpha^{-1}) w_{k,t}^m + \alpha^{-1} w_{k,t}^{\text{md},m} - \gamma g_{k,t}^m$
10: send $Q(w_{k,\tau}^m - w_k), Q(w_{k,\tau}^{\text{ag},m} - w_k^{\text{ag}})$
11: server finds $w_{k+1} \leftarrow w_k + \frac{1}{M} \sum_{m=1}^M Q(w_{k,\tau}^m - w_k),\ w_{k+1}^{\text{ag}} \leftarrow w_k^{\text{ag}} + \frac{1}{M} \sum_{m=1}^M Q(w_{k,\tau}^{\text{ag},m} - w_k^{\text{ag}})$

5 Convergence Analysis

The rigorous theoretical guarantees of reducing communication complexity under strongly-convex and homogeneous assumptions should come first to ensure the significance of FedAQ as one of the standards of communication-efficient federated optimization algorithms. Proving convergence guarantees of FedAQ even under these assumptions requires careful consideration of the approximation error induced by the quantization scheme combined with the convergence analysis of acceleration based methods. To recall, in FedAQ the server aggregates two quantized local updates $Q(w_{k,\tau}^m - w_k), Q(w_{k,\tau}^{\text{ag},m} - w_k^{\text{ag}})$ from all clients (See line 11 in Algorithm 3) in each round. If we simply try to generalize the convergence guarantee of FedAC to incorporate the quantization variance costs, the proof techniques from earlier quantization-based methods cannot be directly applied, as we now have two additional quantization error terms that contribute to the overall cost. A significant amount of additional effort is required in order to account for this new quantization error.

In this section, we first define two condition sets of hyperparameters used for the convergence analysis of FedAQ. Then, we provide the proof sketch of FedAQ under one such condition set that leads to the better convergence rate $\tilde{\mathcal{O}}(\frac{1+q}{MT} + \frac{1+q}{TK^3})$. The full proofs of lemmas, theorems, and corollaries under both condition sets are elaborated in section 5.3 and section 5.4. Finally, we discuss how we obtain the new convergence rate for [9] and look into more theoretical details on contribution 2 in Introduction.

5.1 Two Parameter Condition Sets

We carefully determine two parameter condition sets that theoretically ensure the convergence guarantees. The first one is

$$\eta, \gamma \in \Big(0, \frac{1}{L}\Big], \gamma = \max\Big(\sqrt{\frac{\eta}{\mu\tau}}, \eta\Big), \alpha = \frac{1}{\gamma\mu}, \beta = \alpha + 1 \tag{33}$$

We add one more condition $\gamma \in (0, \frac{1}{L}]$ to the FedAC-I condition [43] and create our parameter condition set (33). The second one is

$$\eta, \gamma \in \Big(0, \frac{1}{L}\Big], \gamma = \max\Big(\sqrt{\frac{\eta}{\mu\tau}}, \eta\Big), \alpha = \frac{3}{2\gamma\mu} - \frac{1}{2}, \beta = \frac{2\alpha^2 - 1}{\alpha - 1}, \gamma\mu \leq \frac{3}{4} \tag{34}$$

We add two more conditions $\gamma \in (0, \frac{1}{L}]$ and $\gamma\mu \leq \frac{3}{4}$ to the FedAC-II condition to build our parameter condition set (34). Even though quantization adds complexity to the algorithm, these weak assumptions are the only

additional requirements for showing the convergence of FedAQ. Moreover, although the better convergence rate $\tilde{\mathcal{O}}(\frac{1+q}{MT}+\frac{1+q}{TK^3})$ is obtained from the condition set (34), we also analyze the convergence of FedAQ under the condition set (33) because this set empirically leads to more stable training and better performance in experiments than the condition set (34) (See Strongly convex case in section 6.2.1). The intuition of the less stable training of FedAQ under the condition set (34) comes from larger α, β than those of the condition set (33). If α, β are too large, α^{-1}, β^{-1} in Algorithm 3 cannot be used as proper coupling coefficients for local parameters $w_{k,t}^m, w_{k,t}^{\text{ag},m}, w_{k,t}^{\text{md},m}$. This results in aggressive updates and less stable training behavior.

5.2 Proof Sketch of FedAQ Under Condition Set (34)

The decentralized potential $\Phi_{k,t}$ [43] is used for our convergence analysis. People commonly use this potential for acceleration analysis [2].

$$\Phi_{k,t} = F(\bar{w}_{k,t}^{\text{ag}}) - F^* + \frac{1}{6}\mu\|\bar{w}_{k,t} - w^*\|^2$$

$\bar{w}_{k,t}$ and $\bar{w}_{k,t}^{\text{ag}}$ is respectively the average of $w_{k,t}^m$ and $w_{k,t}^{\text{ag, m}}$ for all m. Here, we additionally define Φ_k as below.

$$\Phi_k := \Phi_{k,0} = F(w_k^{\text{ag}}) - F^* + \frac{1}{6}\mu\|w_k - w^*\|^2$$

Since w_k and w_k^{ag} are parameters obtained after kth synchronization in a server side, Φ_k can be considered as the potential of server models. Φ_k is essential to show the convergence of FedAQ because there is the computation of the quantizer between $\Phi_{k-1,\tau}$ and $\Phi_{k,0}$. Thus, we should not naively track $\Phi_{k,t}$ but track Φ_k for our analysis. Obtaining $\Phi_k \leq \epsilon$ would imply that $F(w_k^{\text{ag}}) - F^* \leq \epsilon$ and since $F^* \leq F(w_k^{\text{ag}})$, it would also imply that $\|w_k - w^*\|^2 = O(\epsilon)$, thus obtaining convergence in terms of both the objective value and the iterate.

Our goal is to show the convergence of FedAQ and derive the simplified convergence rate so that we can get the number of communication rounds to achieve a linear speedup in M. As the first step to show this, we prove Lemma 5 which represents the relationship between two consecutive server potential functions Φ_k and Φ_{k+1}. The quantization scheme amplifies the instability to the convergence of FedAQ in addition to the effect of acceleration. Despite this challenge, we derive Lemma 5 with the help of subtle Propositions (See section 5.4.1).

Lemma 5: Let F be μ-strongly convex, and assume Assumption 1, 2, 3, 4, then for $\alpha = \frac{3}{2\gamma\mu} - \frac{1}{2}, \beta = \frac{2\alpha^2-1}{\alpha-1}, \gamma \in [\eta, \sqrt{\frac{\eta}{\mu}}], \eta, \gamma \in (0, \frac{1}{L}], \gamma\mu \leq \frac{3}{4}, \tau \geq 2$, FedAQ yields

$$\mathbb{E}[\Phi_{k+1}] \leq D(\gamma,\tau)\mathbb{E}[\Phi_k] + (\frac{\eta^2 L}{2} + \frac{\gamma^2\mu}{6})\frac{\tau\sigma^2}{M} + \gamma\tau \cdot \max_{0\leq t<\tau} \mathbb{E}[\|\nabla F(\bar{w}_{k,t}^{\text{md}}) - \frac{1}{M}\sum_{m=1}^{M} \nabla F(w_{k,t}^{\text{md},m})\|^2]$$
$$+ \underbrace{\frac{q}{M}(\frac{\gamma^2\mu}{3} + \eta^2 L)\tau\sigma^2 + \frac{q}{2M}\Big((\gamma-\eta)^2\gamma^2\mu^2(\frac{\mu}{3}+\frac{L}{4}) + \gamma^4(\frac{\mu}{3}+L)^2 L\Big)\tau^3\sigma^2}_{\text{additional terms due to quantization}}$$

Where $D(\gamma,\tau)$ is defined as

$$D(\gamma,\tau) = (1 - \frac{1}{3}\gamma\mu)^\tau + \underbrace{\frac{q}{M}\Big(\gamma^2\mu(\frac{8}{3}\mu + 2L) + 2\gamma^2 L(\frac{\mu}{3}+L)\Big)\tau^2}_{\text{additional terms due to quantization}}$$

We get the inequality between Φ_k and Φ_{k+1} by finding the upper bounds of error terms due to multiple(τ) local steps and the quantization step. The upper bound of the error caused by multiple local steps is obtained

with the help of the analysis in [43] (See Proposition 19). Also, we get the tight upper bound of the error due to quantization with our new proof techniques (See Proposition 20, 21, 22). The key challenge in bounding the quantization error terms comes from representing the upper bound of variances of the quantizer Q on two local updates $w_{k,\tau}^m - w_k, w_{k,\tau}^{\text{ag},m} - w_k^{\text{ag}}$ in the form of a server potential Φ_k. Some terms in Lemma 5 are similar to those in Lemma C.2 of the FedAC paper [43], but our lemma contains additional terms that emerge from the quantization scheme.

For the next step, by telescoping Lemma 5, we obtain the main theoretical result Theorem 6. Theorem 6 represents how Φ_K decreases from the initial potential Φ_0 as a communication round K increases. Since we aim to telescope Lemma 5, $D(\gamma, \tau)$ should be smaller than 1. Specifically, we show $D(\gamma, \tau) \leq 1 - \frac{1}{6}\gamma\mu\tau$ with condition (35) (See section 5.4.2). That's why Theorem 6 requires the learning rate γ to satisfy the certain condition (35).

Theorem 6: Let F be μ-strongly convex, and assume Assumption 1, 2, 3, 4, then for the parameter condition set (34), $\tau \geq 2$, if the learning rate γ satisfies

$$\left(\frac{1}{9}\mu^2 + \frac{q}{M}\Big(\mu(\frac{8}{3}\mu + 2L) + 2L(\frac{\mu}{3} + L)\Big)\right)\gamma\tau \leq \frac{1}{6}\mu \tag{35}$$

FedAQ yields

$$\begin{aligned}\mathbb{E}[\Phi_K] \leq \exp\Big(-\frac{1}{6}\max(\eta\mu, \sqrt{\frac{\eta\mu}{\tau}})K\tau\Big)\Phi_0 + \frac{2(2q+1)\eta^{\frac{1}{2}}\sigma^2}{\mu^{\frac{1}{2}}M\tau^{\frac{1}{2}}} + \frac{8(q+25)\eta^2L^2\tau\sigma^2}{\mu} \\ + \frac{3q\Big(\mu^2(\frac{\mu}{3} + \frac{L}{4}) + L(\frac{\mu}{3} + L)^2\Big)\eta^{\frac{3}{2}}\tau^{\frac{1}{2}}\sigma^2}{\mu^{\frac{5}{2}}M} + \frac{3qL(\frac{\mu}{3} + L)^2\eta^3\tau^2\sigma^2}{\mu M}\end{aligned}$$

We get the convergence rate of FedAQ with respect to η under the condition set (34). The final step is to tune η appropriately and obtain a more intuitive form of convergence rate that we can easily analyze a linear speedup in M. The exact form of this can be found in Corollary 23. Here, we introduce the simplified form of Corollary 23.

Corollary 7: (Simplified form of Corollary 23) Note that $T = K\tau$. For $\eta = \min(\frac{1}{L}, \tilde{\Theta}(\frac{\tau}{\mu T^2}))$, FedAQ yields

$$\mathbb{E}[\Phi_K] \leq \min\Big(\exp(-\frac{\mu T}{6L}), \exp(-\frac{\mu^{\frac{1}{2}}T}{6L^{\frac{1}{2}}\tau^{\frac{1}{2}}})\Big)\Phi_0 + \tilde{\mathcal{O}}(\underbrace{\frac{(1+q)\sigma^2}{\mu MT}}_{\text{I}} + \underbrace{\frac{(1+q)L^2\tau^3\sigma^2}{\mu^3T^4}}_{\text{II}} + \underbrace{\frac{qL^3\tau^2\sigma^2}{\mu^4MT^3}}_{\text{III}})$$

The convergence rate of FedAQ under the condition set (33) is obtained in a similar way. The convergence analysis under the condition set (33) is elaborated as Lemma 10, Theorem 16, and Corollary 17 in section 5.3.

Remark 8: The above convergence rate is worse than the convergence rate of FedAC-II according to Theorem C.13 in [43] because there are additive terms related to the quantization noise q in our case. Let's figure out the dominant terms with $\tilde{\mathcal{O}}$ notation from the above convergence rate. Here, we replace τ with $\frac{T}{K}$. At first, we can ignore the first term because it decreases exponentially. The second term I would be $\tilde{\mathcal{O}}(\frac{1+q}{MT})$. Then, the third term II becomes $\tilde{\mathcal{O}}(\frac{(1+q)\tau^3}{T^4}) = \tilde{\mathcal{O}}(\frac{1+q}{TK^3})$. Finally, the last term III turns into $\tilde{\mathcal{O}}(\frac{q\tau^2}{MT^3}) = \tilde{\mathcal{O}}(\frac{q}{MTK^2})$. Thus, the overall convergence rate of FedAQ under the condition set (34) would be $\tilde{\mathcal{O}}(\frac{1+q}{MT} + \frac{1+q}{TK^3})$. Similarly, we obtain the simplified convergence rate of FedAQ under the condition set (33) from three terms (14), (15), (16) of Corollary 17. In this case, the convergence rate of FedAQ is $\tilde{\mathcal{O}}(\frac{1+q}{MT} + \frac{1}{TK^2})$, and the required number of communication rounds to achieve a linear speedup in M is $\tilde{\mathcal{O}}((\frac{M}{1+q})^{\frac{1}{2}})$.

Remark 9: As we mention above, FedAQ converges at rate $\tilde{\mathcal{O}}(\frac{1+q}{MT}+\frac{1+q}{TK^3})$, which is better than the convergence rate of [9] $\tilde{\mathcal{O}}(\frac{1+q}{MT}+\frac{1}{TK})$. To our knowledge, [9] obtain the best convergence rate among previous quantization-based federated optimization algorithms. Actually, in the strongly-convex and homogeneous case, [9] provide different convergence rate $\mathcal{O}(\frac{1}{\gamma^2\tau}+\frac{(q+1)}{(\frac{q}{M}+1)\tau M})=\mathcal{O}(\frac{K}{\gamma^2 T}+\frac{(q+1)K}{(\frac{q}{M}+1)TM})$, where γ is a learning rate for the server updates. They achieve this convergence rate by tuning $\eta=\frac{1}{2L(\frac{q}{M}+1)\tau\gamma}$. However, we cannot say this algorithm achieves a linear speedup in this scenario. That's why we provide a new convergence rate $\tilde{\mathcal{O}}(\frac{1+q}{MT}+\frac{1}{TK})$ for [9] by tuning η in a different way. This new η makes this algorithm achieve a linear speedup. Why the original η cannot achieve a linear speedup and how we get new η can be found in section 5.5.

5.3 Proof Details for FedAQ under Condition Set (33)

Before diving into proof details, we define $\bar{w}_{k,\tau}, \bar{w}^{\text{ag}}_{k,\tau}, \Psi^m_{k,t}, \Psi_{k,t}, \Psi_k, A^m_{k,t}$ as below.

$$
\begin{aligned}
\bar{w}_{k,\tau} &= \frac{1}{M}\sum_{m=1}^{M} w^m_{k,\tau} \\
\bar{w}^{\text{ag}}_{k,\tau} &= \frac{1}{M}\sum_{m=1}^{M} w^{\text{ag},m}_{k,\tau} \\
\Psi^m_{k,t} &= F(w^{\text{ag},m}_{k,t}) - F^* + \frac{1}{2}\mu\|w^m_{k,t} - w^*\|^2 \\
\Psi_{k,t} &= \frac{1}{M}\sum_{m=1}^{M} F(w^{\text{ag},m}_{k,t}) - F^* + \frac{1}{2}\mu\|\bar{w}_{k,t} - w^*\|^2 \\
\Psi_k :&= \Psi_{k,0} = F(w^{\text{ag}}_k) - F^* + \frac{1}{2}\mu\|w_k - w^*\|^2 \\
A^m_{k,t} &= \frac{\gamma^2\mu^2(\mu+L)}{(1+\gamma\mu)^2}\|w^m_{k,t} - w^{\text{ag},m}_{k,t}\|^2 + \gamma^2(\mu+L)\frac{2L}{1+\gamma\mu}\Psi^m_{k,t}
\end{aligned}
$$

The above notations are essential to our convergence analysis. Intuitively, if the FedAQ algorithm converges to the optimal point, $\bar{w}_{k,\tau}, \bar{w}^{\text{ag}}_{k,\tau}$ become w^*, and $\Psi^m_{k,t}, \Psi_{k,t}, \Psi_k, A^m_{k,t}$ become 0. In order to denote the σ-algebra generated by $\{w^m_{k',t'}, w^{\text{ag},m}_{k',t'}\}_{(k'<k) \text{ or } (k'=k,t'\le t), m\in[M]}$, we use $\mathcal{F}_{k,t}$.

5.3.1 Proof of Lemma 10

Lemma 10: Let F be μ-strongly convex, and assume Assumption 1, 2, 3, 4, then for $\alpha=\frac{1}{\gamma\mu}, \beta=\alpha+1, \gamma\in[\eta,\sqrt{\frac{\eta}{\mu}}], \eta,\gamma\in(0,\frac{1}{L}], \tau\ge 2$, FedAQ yields

$$
\begin{aligned}
\mathbb{E}[\Psi_{k+1}] \le\ & C(\gamma,\tau)\mathbb{E}[\Psi_k] + \frac{1}{2}(\eta^2 L + \frac{\gamma^2\mu}{M})\tau\sigma^2 \\
& + \gamma\mu L\tau\cdot\max_{0\le t<\tau}\mathbb{E}[\frac{1}{M}\sum_{m=1}^{M}\|\bar{w}^{\text{md}}_{k,t} - w^{\text{md},m}_{k,t}\|\|\frac{1}{1+\gamma\mu}(\bar{w}_{k,t} - w^m_{k,t}) + \frac{\gamma\mu}{1+\gamma\mu}(\bar{w}^{\text{ag}}_{k,t} - w^{\text{ag},m}_{k,t})\|] \\
& + \underbrace{\frac{q}{M}(\gamma^2\mu + \eta^2 L)\tau\sigma^2 + \frac{q}{2M}\Big(\frac{(\gamma-\eta)^2\gamma^2\mu^2(\mu+L)}{(1+\gamma\mu)^2} + \frac{\gamma^4(\mu+L)^2L}{1+\gamma\mu}\Big)\tau^3\sigma^2}_{\text{Additional terms due to quantization}}
\end{aligned}
$$

Where $C(\gamma, \tau)$ is defined as

$$C(\gamma,\tau) = (1-\gamma\mu)^\tau + \underbrace{\frac{q}{M}\Big(\frac{4\gamma^2\mu(\mu+L)}{(1+\gamma\mu)^2} + \frac{2L\gamma^2(\mu+L)}{1+\gamma\mu}\Big)\tau^2}_{\text{Additional terms due to quantization}}$$

In this section, we first introduce five crucial Propositions for proving Lemma 10. Then, we prove Lemma 10 by using Propositions in the last part of this section.

Proposition 11: Let Assumption 1 hold and consider any k synchronization round. Then, we can decompose the expectation as follows:

$$\mathbb{E}[\|w_{k+1} - w^*\|^2] = \mathbb{E}[\|w_{k+1} - \bar{w}_{k,\tau}\|^2] + \mathbb{E}[\|\bar{w}_{k,\tau} - w^*\|^2]$$
$$\mathbb{E}[F(w_{k+1}^{\text{ag}}) - F^*] = \mathbb{E}[F(w_{k+1}^{\text{ag}}) - \frac{1}{M}\sum_{m=1}^{M} F(w_{k,\tau}^{\text{ag},m})] + \mathbb{E}[\frac{1}{M}\sum_{m=1}^{M} F(w_{k,\tau}^{\text{ag},m}) - F^*]$$

Proof of Proposition 11 The second equality is trivial. Let's focus on the first equality. By Assumption 1, the quantizer Q is unbiased and we get,

$$\mathbb{E}_Q[w_{k+1}] = w_k + \frac{1}{M}\sum_{m=1}^{M}\mathbb{E}_Q Q(w_{k,\tau}^m - w_k) = \frac{1}{M}\sum_{m=1}^{M} w_{k,\tau}^m = \bar{w}_{k,\tau}$$

Thus, we finally obtain

$$\begin{aligned}\mathbb{E}[\|w_{k+1} - w^*\|^2] &= \mathbb{E}[\|w_{k+1} - \bar{w}_{k,\tau} + \bar{w}_{k,\tau} - w^*\|^2] \\ &= \mathbb{E}[\|w_{k+1} - \bar{w}_{k,\tau}\|^2] + \mathbb{E}[\|\bar{w}_{k,\tau} - w^*\|^2]\end{aligned}$$

Proposition 12: Let F be μ-strongly convex, and assume Assumption 2, 3, 4, then for $\alpha = \frac{1}{\gamma\mu}, \beta = \alpha + 1, \gamma \in [\eta, \sqrt{\frac{\eta}{\mu}}], \eta \in (0, \frac{1}{L}]$, FedAQ yields

$$\mathbb{E}[\Psi_{k,\tau}] \le (1-\gamma\mu)^\tau \mathbb{E}[\Psi_k] + \frac{1}{2}(\eta^2 L + \frac{\gamma^2\mu}{M})\tau\sigma^2 + \gamma\mu L\tau$$
$$\cdot \max_{0\le t<\tau} \mathbb{E}[\frac{1}{M}\sum_{m=1}^{M}\|\bar{w}_{k,t}^{\text{md}} - w_{k,t}^{\text{md},m}\|\|\frac{1}{1+\gamma\mu}(\bar{w}_{k,t} - w_{k,t}^m) + \frac{\gamma\mu}{1+\gamma\mu}(\bar{w}_{k,t}^{\text{ag}} - w_{k,t}^{\text{ag},m})\|]$$

Proof of Proposition 12 We refer to the proof of Lemma B.2 in [43]. There is no quantization between $\Psi_{k,\tau}$ and Ψ_k. Thus, we can directly apply useful inequalities in the proof of Lemma B.2 in [43] to our proof. Then, we obtain

$$\mathbb{E}[\Psi_{k,t+1}|\mathcal{F}_{k,t}] \le (1-\gamma\mu)\Psi_{k,t} + \frac{1}{2}(\eta^2 L + \frac{\gamma^2\mu}{M})\sigma^2 + \gamma\mu L$$
$$\cdot \frac{1}{M}\sum_{m=1}^{M}\|\bar{w}_{k,t}^{\text{md}} - w_{k,t}^{\text{md},m}\|\|\frac{1}{1+\gamma\mu}(\bar{w}_{k,t} - w_{k,t}^m) + \frac{\gamma\mu}{1+\gamma\mu}(\bar{w}_{k,t}^{\text{ag}} - w_{k,t}^{\text{ag},m})\|$$

From the above relationship between $\Psi_{k,t+1}$ and $\Psi_{k,t}$, we get

$$\begin{aligned}\mathbb{E}[\Psi_{k,\tau}] &\leq (1-\gamma\mu)^{\tau}\mathbb{E}[\Psi_k] + \Big(\sum_{t=0}^{\tau-1}(1-\gamma\mu)^t\Big)\frac{1}{2}(\eta^2 L + \frac{\gamma^2\mu}{M})\sigma^2 + \gamma\mu L \cdot \sum_{t=0}^{\tau-1}\Big\{(1-\gamma\mu)^{\tau-t-1} \\ &\mathbb{E}[\frac{1}{M}\sum_{m=1}^{M}\|\bar{w}_{k,t}^{\text{md}} - w_{k,t}^{\text{md},m}\|\|\frac{1}{1+\gamma\mu}(\bar{w}_{k,t} - w_{k,t}^m) + \frac{\gamma\mu}{1+\gamma\mu}(\bar{w}_{k,t}^{\text{ag}} - w_{k,t}^{\text{ag},m})\|]\Big\} \\ &\leq (1-\gamma\mu)^{\tau}\mathbb{E}[\Psi_k] + \frac{1}{2}(\eta^2 L + \frac{\gamma^2\mu}{M})\tau\sigma^2 + \gamma\mu L\tau \\ &\cdot \max_{0\leq t<\tau}\mathbb{E}[\frac{1}{M}\sum_{m=1}^{M}\|\bar{w}_{k,t}^{\text{md}} - w_{k,t}^{\text{md},m}\|\|\frac{1}{1+\gamma\mu}(\bar{w}_{k,t} - w_{k,t}^m) + \frac{\gamma\mu}{1+\gamma\mu}(\bar{w}_{k,t}^{\text{ag}} - w_{k,t}^{\text{ag},m})\|]\end{aligned}$$

Proposition 13: Let Assumption 1 hold. Then, we have

$$\begin{aligned}\mathbb{E}[\|w_{k+1} - \bar{w}_{k,\tau}\|^2] &\leq \frac{q}{M^2}\sum_{m=1}^{M}\mathbb{E}[\|w_{k,\tau}^m - w_k\|^2] \\ \mathbb{E}[F(w_{k+1}^{\text{ag}}) - \frac{1}{M}\sum_{m=1}^{M}F(w_{k,\tau}^{\text{ag},m})] &\leq \frac{qL}{2M^2}\sum_{m=1}^{M}\mathbb{E}[\|w_{k,\tau}^{\text{ag},m} - w_k^{\text{ag}}\|^2]\end{aligned}$$

Proof of Proposition 13 First, let's consider the first inequality. According to Assumption 1, we get

$$\begin{aligned}\mathbb{E}[\|w_{k+1} - \bar{w}_{k,\tau}\|^2] &= \mathbb{E}[\|w_k + \frac{1}{M}\sum_{m=1}^{M}Q(w_{k,\tau}^m - w_k) - \frac{1}{M}\sum_{m=1}^{M}w_{k,\tau}^m\|^2] \\ &= \mathbb{E}[\|\frac{1}{M}\sum_{m=1}^{M}Q(w_{k,\tau}^m - w_k) - (w_{k,\tau}^m - w_k)\|^2] \\ &= \frac{1}{M^2}\sum_{m=1}^{M}\mathbb{E}[\|Q(w_{k,\tau}^m - w_k) - (w_{k,\tau}^m - w_k)\|^2] \leq \frac{q}{M^2}\sum_{m=1}^{M}\mathbb{E}\|w_{k,\tau}^m - w_k\|^2\end{aligned}$$

The third equality comes from the unbiasedness of Q, and the last inequality stems from the variance assumption of Q. Similarly, we obtain

$$\begin{aligned}
\mathbb{E}[F(w_{k+1}^{\text{ag}}) - \frac{1}{M}\sum_{m=1}^{M} F(w_{k,\tau}^{\text{ag},m})] &= \mathbb{E}[F(w_k^{\text{ag}} + \frac{1}{M}\sum_{m=1}^{M} Q(w_{k,\tau}^{\text{ag},m} - w_k^{\text{ag}})) - \frac{1}{M}\sum_{m=1}^{M} F(w_{k,\tau}^{\text{ag},m})] \\
&= \mathbb{E}[\frac{1}{M}\sum_{m=1}^{M} F(w_k^{\text{ag}} + \frac{1}{M}\sum_{m=1}^{M} Q(w_{k,\tau}^{\text{ag},m} - w_k^{\text{ag}})) - F(w_{k,\tau}^{\text{ag},m})] \\
&\leq \mathbb{E}\Big[\frac{1}{M}\sum_{m=1}^{M}\langle \nabla F(w_{k,\tau}^{\text{ag},m}), \frac{1}{M}\sum_{m=1}^{M}\Big(Q(w_{k,\tau}^{\text{ag},m} - w_k^{\text{ag}}) - (w_{k,\tau}^{\text{ag},m} \\
&\quad - w_k^{\text{ag}})\Big)\rangle + \frac{L}{2}\|\frac{1}{M}\sum_{m=1}^{M} Q(w_{k,\tau}^{\text{ag},m} - w_k^{\text{ag}}) - (w_{k,\tau}^{\text{ag},m} - w_k^{\text{ag}})\|^2\Big] \\
&= \frac{L}{2}\mathbb{E}[\|\frac{1}{M}\sum_{m=1}^{M} Q(w_{k,\tau}^{\text{ag},m} - w_k^{\text{ag}}) - (w_{k,\tau}^{\text{ag},m} - w_k^{\text{ag}})\|^2] \\
&= \frac{L}{2M^2}\sum_{m=1}^{M}\mathbb{E}[\|Q(w_{k,\tau}^{\text{ag},m} - w_k^{\text{ag}}) - (w_{k,\tau}^{\text{ag},m} - w_k^{\text{ag}})\|^2] \\
&\leq \frac{qL}{2M^2}\sum_{m=1}^{M}\mathbb{E}[\|w_{k,\tau}^{\text{ag},m} - w_k^{\text{ag}}\|^2]
\end{aligned}$$

Proposition 14: Let F be μ-strongly convex, and assume Assumption 2, 3, 4, then for $\alpha = \frac{1}{\gamma\mu}, \beta = \alpha + 1, \gamma \in [\eta, \sqrt{\frac{\eta}{\mu}}], \eta, \gamma \in (0, \frac{1}{L}]$, we get

$$\mathbb{E}[A_{k,t}^m] \leq \mathbb{E}[A_{k,0}^m] + \Big(\frac{(\gamma-\eta)^2(\mu+L)}{1+\gamma\mu} + \frac{\gamma^2(\mu+L)^2 L}{\mu^2}\Big)\cdot\Big(1 - (1 - \gamma\mu + \frac{\gamma\mu}{1+\gamma\mu})^t\Big)\sigma^2$$

Proof of Proposition 14 From the notation mentioned in the beginning of section 5.3,

$$\mathbb{E}[A_{k,t+1}^m|\mathcal{F}_{k,t}] = \frac{\gamma^2\mu^2(\mu+L)}{(1+\gamma\mu)^2}\mathbb{E}[\|w_{k,t+1}^m - w_{k,t+1}^{\text{ag},m}\|^2|\mathcal{F}_{k,t}] + \gamma^2(\mu+L)\frac{2L}{1+\gamma\mu}\mathbb{E}[\Psi_{k,t+1}^m|\mathcal{F}_{k,t}] \tag{36}$$

Thus, let's sequentially compute $\mathbb{E}[\|w_{k,t+1}^m - w_{k,t+1}^{\text{ag},m}\|^2|\mathcal{F}_{k,t}]$ and $\mathbb{E}[\Psi_{k,t+1}^m|\mathcal{F}_{k,t}]$.

$$\begin{aligned}
\mathbb{E}[\|w_{k,t+1}^m - w_{k,t+1}^{\text{ag},m}\|^2|\mathcal{F}_{k,t}] &= \mathbb{E}[\|(1-\alpha^{-1})w_{k,t}^m + \alpha^{-1}w_{k,t}^{\text{md},m} - \gamma g_{k,t}^m - w_{k,t}^{\text{md},m} + \eta g_{k,t}^m\|^2|\mathcal{F}_{k,t}] \\
&= \mathbb{E}[\|(1-\alpha^{-1})(w_{k,t}^m - w_{k,t}^{\text{md},m}) - (\gamma-\eta)g_{k,t}^m\|^2|\mathcal{F}_{k,t}]\ (\leftarrow \gamma \geq \eta) \\
&= \|(1-\alpha^{-1})(w_{k,t}^m - w_{k,t}^{\text{md},m}) - (\gamma-\eta)\nabla F(w_{k,t}^{\text{md},m})\|^2 \\
&\quad + (\gamma-\eta)^2\mathbb{E}[\|\nabla F(w_{k,t}^{\text{md},m}) - g_{k,t}^m\|^2|\mathcal{F}_{k,t}] \\
&\leq (1-\alpha^{-1})^2\|w_{k,t}^m - w_{k,t}^{\text{md},m}\|^2 + (\gamma-\eta)^2\|\nabla F(w_{k,t}^{\text{md},m})\|^2 \\
&\quad + (\gamma-\eta)^2\sigma^2 - 2(\gamma-\eta)\langle(1-\alpha^{-1})(w_{k,t}^m - w_{k,t}^{\text{md},m}), \nabla F(w_{k,t}^{\text{md},m})\rangle \\
&\leq (1-\alpha^{-1})^2(1+\gamma\mu)\|w_{k,t}^m - w_{k,t}^{\text{md},m}\|^2 \\
&\quad + (\gamma-\eta)^2(1+\frac{1}{\gamma\mu})\|\nabla F(w_{k,t}^{\text{md},m})\|^2 + (\gamma-\eta)^2\sigma^2 \\
&= \frac{(1-\gamma\mu)^2}{1+\gamma\mu}\|w_{k,t}^m - w_{k,t}^{\text{ag},m}\|^2 + (\gamma-\eta)^2\frac{1+\gamma\mu}{\gamma\mu}\|\nabla F(w_{k,t}^{\text{md},m})\|^2 + (\gamma-\eta)^2\sigma^2
\end{aligned}$$

Here, we need to bound $\|\nabla F(w_{k,t}^{\text{md},m})\|^2$.

$$\begin{aligned}
\|\nabla F(w_{k,t}^{\text{md},m})\|^2 &\leq 2L(F(w_{k,t}^{\text{md},m}) - F^*) \ (\because \text{ Assumption 3}) \\
&\leq 2L\Big(\beta^{-1}(F(w_{k,t}^{m}) - F(w^*)) + (1-\beta^{-1})(F(w_{k,t}^{\text{ag},m}) - F^*)\Big) \\
&\leq \beta^{-1}L^2\|w_{k,t}^{m} - w^*\|^2 + 2(1-\beta^{-1})L(F(w_{k,t}^{\text{ag},m}) - F^*) \\
&= \frac{\gamma\mu L^2}{1+\gamma\mu}\|w_{k,t}^{m} - w^*\|^2 + \frac{2L}{1+\gamma\mu}(F(w_{k,t}^{\text{ag},m}) - F^*) \\
&\leq \frac{\mu L}{1+\gamma\mu}\|w_{k,t}^{m} - w^*\|^2 + \frac{2L}{1+\gamma\mu}(F(w_{k,t}^{\text{ag},m}) - F^*) = \frac{2L}{1+\gamma\mu}\Psi_{k,t}^{m}
\end{aligned} \tag{37}$$

The last inequality comes from the fact $\gamma \in [0, \frac{1}{L})$. Therefore, we finally get

$$\begin{aligned}
\mathbb{E}[\|w_{k,t+1}^{m} - w_{k,t+1}^{\text{ag},m}\|^2|\mathcal{F}_{k,t}] &\leq \frac{(1-\gamma\mu)^2}{1+\gamma\mu}\|w_{k,t}^{m} - w_{k,t}^{\text{ag},m}\|^2 + (\gamma-\eta)^2\frac{1+\gamma\mu}{\gamma\mu}\|\nabla F(w_{k,t}^{\text{md},m})\|^2 + (\gamma-\eta)^2\sigma^2 \\
&\leq \frac{(1-\gamma\mu)^2}{1+\gamma\mu}\|w_{k,t}^{m} - w_{k,t}^{\text{ag},m}\|^2 + (\gamma-\eta)^2\frac{1+\gamma\mu}{\gamma\mu}\Big(\frac{2L}{1+\gamma\mu}\Psi_{k,t}^{m}\Big) + (\gamma-\eta)^2\sigma^2
\end{aligned} \tag{38}$$

Now, let's compute $\mathbb{E}[\Psi_{k,t+1}^{m}|\mathcal{F}_{k,t}]$. We need to compute $\mathbb{E}[\|w_{k,t+1}^{m} - w^*\|^2|\mathcal{F}_{k,t}]$ and $\mathbb{E}[F(w_{k,t+1}^{\text{ag},m}) - F^*|\mathcal{F}_{k,t}]$ first.

$$\begin{aligned}
\mathbb{E}[\|w_{k,t+1}^{m} - w^*\|^2|\mathcal{F}_{k,t}] &= \mathbb{E}[\|(1-\alpha^{-1})w_{k,t}^{m} + \alpha^{-1}w_{k,t}^{\text{md},m} - \gamma g_{k,t}^{m} - w^*\|^2|\mathcal{F}_{k,t}] \\
&\leq \|(1-\alpha^{-1})w_{k,t}^{m} + \alpha^{-1}w_{k,t}^{\text{md},m} - w^*\|^2 + \gamma^2\|\nabla F(w_{k,t}^{\text{md},m})\|^2 + \gamma^2\sigma^2 \\
&\quad - 2\gamma\langle(1-\alpha^{-1})w_{k,t}^{m} + \alpha^{-1}w_{k,t}^{\text{md},m} - w^*, \nabla F(w_{k,t}^{\text{md},m})\rangle \\
&\leq (1-\alpha^{-1})\|w_{k,t}^{m} - w^*\|^2 + \alpha^{-1}\|w_{k,t}^{\text{md},m} - w^*\|^2 + \gamma^2\|\nabla F(w_{k,t}^{\text{md},m})\|^2 + \gamma^2\sigma^2 \\
&\quad - 2\gamma\langle(1-\alpha^{-1}(1-\beta^{-1}))w_{k,t}^{m} + \alpha^{-1}(1-\beta^{-1})w_{k,t}^{\text{ag},m} - w^*, \nabla F(w_{k,t}^{\text{md},m})\rangle \\
&= (1-\gamma\mu)\|w_{k,t}^{m} - w^*\|^2 + \gamma\mu\|w_{k,t}^{\text{md},m} - w^*\|^2 + \gamma^2\|\nabla F(w_{k,t}^{\text{md},m})\|^2 + \gamma^2\sigma^2 \\
&\quad - 2\gamma\langle\frac{1}{1+\gamma\mu}w_{k,t}^{m} + \frac{\gamma\mu}{1+\gamma\mu}w_{k,t}^{\text{ag},m} - w^*, \nabla F(w_{k,t}^{\text{md},m})\rangle
\end{aligned}$$

$$\begin{aligned}
&\mathbb{E}[F(w_{k,t+1}^{\text{ag},m}) - F^*|\mathcal{F}_{k,t}] \\
&\quad\leq \mathbb{E}[F(w_{k,t}^{\text{md},m}) + \langle\nabla F(w_{k,t}^{\text{md},m}), w_{k,t+1}^{\text{ag},m} - w_{k,t}^{\text{md},m}\rangle + \frac{L}{2}\|w_{k,t+1}^{\text{ag},m} - w_{k,t}^{\text{md},m}\|^2 - F^*|\mathcal{F}_{k,t}] \\
&\quad\leq F(w_{k,t}^{\text{md},m}) - F^* - \eta\|\nabla F(w_{k,t}^{\text{md},m})\|^2 + \frac{\eta^2 L}{2}\|\nabla F(w_{k,t}^{\text{md},m})\|^2 + \frac{\eta^2 L}{2}\sigma^2 \\
&\quad\leq F(w_{k,t}^{\text{md},m}) - F^* - \frac{\eta}{2}\|\nabla F(w_{k,t}^{\text{md},m})\|^2 + \frac{\eta^2 L}{2}\sigma^2 \ (\because 1 - \frac{\eta L}{2} \geq \frac{1}{2} \leftarrow \eta \in [0, \frac{1}{L}]) \\
&\quad= (1-\alpha^{-1})(F(w_{k,t}^{\text{ag},m}) - F^*) + \alpha^{-1}(F(w_{k,t}^{\text{md},m}) - F^*) \\
&\quad+ (1-\alpha^{-1})(F(w_{k,t}^{\text{md},m}) - F(w_{k,t}^{\text{ag},m})) - \frac{\eta}{2}\|\nabla F(w_{k,t}^{\text{md},m})\|^2 + \frac{\eta^2 L}{2}\sigma^2
\end{aligned}$$

$$
\begin{aligned}
&\leq (1-\alpha^{-1})(F(w_{k,t}^{\mathrm{ag},m}) - F^*) - \frac{\mu\alpha^{-1}}{2}\|w_{k,t}^{\mathrm{md},m} - w^*\|^2 + \alpha^{-1}\langle \nabla F(w_{k,t}^{\mathrm{md},m}), w_{k,t}^{\mathrm{md},m} - w^*\rangle \\
&+ (1-\alpha^{-1})\langle \nabla F(w_{k,t}^{\mathrm{md},m}), w_{k,t}^{\mathrm{md},m} - w_{k,t}^{\mathrm{ag},m}\rangle - \frac{\eta}{2}\|\nabla F(w_{k,t}^{\mathrm{md},m})\|^2 + \frac{\eta^2 L}{2}\sigma^2 \\
&= (1-\alpha^{-1})(F(w_{k,t}^{\mathrm{ag},m}) - F^*) - \frac{\mu\alpha^{-1}}{2}\|w_{k,t}^{\mathrm{md},m} - w^*\|^2 - \frac{\eta}{2}\|\nabla F(w_{k,t}^{\mathrm{md},m})\|^2 + \frac{\eta^2 L}{2}\sigma^2 \\
&+ \alpha^{-1}\langle \nabla F(w_{k,t}^{\mathrm{md},m}), \alpha\beta^{-1}w_{k,t}^m + (1-\alpha\beta^{-1})w_{k,t}^{\mathrm{ag},m} - w^*\rangle \\
&= (1-\gamma\mu)(F(w_{k,t}^{\mathrm{ag},m}) - F^*) - \frac{\gamma\mu^2}{2}\|w_{k,t}^{\mathrm{md},m} - w^*\|^2 - \frac{\eta}{2}\|\nabla F(w_{k,t}^{\mathrm{md},m})\|^2 + \frac{\eta^2 L}{2}\sigma^2 \\
&+ \gamma\mu\langle \frac{1}{1+\gamma\mu}w_{k,t}^m + \frac{\gamma\mu}{1+\gamma\mu}w_{k,t}^{\mathrm{ag},m} - w^*, \nabla F(w_{k,t}^{\mathrm{md},m})\rangle
\end{aligned}
$$

Then, we bound $\mathbb{E}[\Psi_{k,t+1}^m|\mathcal{F}_{k,t}]$ by using the above results.

$$
\begin{aligned}
\mathbb{E}[\Psi_{k,t+1}^m|\mathcal{F}_{k,t}] &= \frac{\mu}{2}\mathbb{E}[\|w_{k,t+1}^m - w^*\|^2|\mathcal{F}_{k,t}] + \mathbb{E}[F(w_{k,t+1}^{\mathrm{ag},m}) - F^*|\mathcal{F}_{k,t}] \\
&\leq (1-\gamma\mu)\Psi_{k,t}^m - \frac{\eta - \gamma^2\mu}{2}\|\nabla F(w_{k,t}^{\mathrm{md},m})\|^2 + \frac{\gamma^2\mu + \eta^2 L}{2}\sigma^2 \\
&\leq (1-\gamma\mu)\Psi_{k,t}^m + \frac{\gamma^2\mu + \eta^2 L}{2}\sigma^2 \ (\because \gamma \leq \sqrt{\frac{\eta}{\mu}}) \\
&\leq (1-\gamma\mu)\Psi_{k,t}^m + \frac{\gamma^2(\mu + L)}{2}\sigma^2
\end{aligned} \tag{39}
$$

Plugging (38), (39) in (36) yields,

$$
\begin{aligned}
&\mathbb{E}[A_{k,t+1}^m|\mathcal{F}_{k,t}] \\
&\leq \frac{\gamma^2\mu^2(\mu+L)}{(1+\gamma\mu)^2}\Big(\frac{(1-\gamma\mu)^2}{1+\gamma\mu}\|w_{k,t}^m - w_{k,t}^{\mathrm{ag},m}\|^2 + (\gamma-\eta)^2\frac{1+\gamma\mu}{\gamma\mu}\Big(\frac{2L}{1+\gamma\mu}\Psi_{k,t}^m\Big) + (\gamma-\eta)^2\sigma^2\Big) \\
&+ \gamma^2(\mu+L)\frac{2L}{1+\gamma\mu}\Big((1-\gamma\mu)\Psi_{k,t}^m + \frac{\gamma^2(\mu+L)}{2}\sigma^2\Big) \\
&= \frac{(1-\gamma\mu)^2}{1+\gamma\mu}\cdot\frac{\gamma^2\mu^2(\mu+L)}{(1+\gamma\mu)^2}\|w_{k,t}^m - w_{k,t}^{\mathrm{ag},m}\|^2 + \Big(\frac{\gamma\mu(\gamma-\eta)^2(\mu+L)}{1+\gamma\mu} + \gamma^2(\mu+L)(1-\gamma\mu)\Big)\frac{2L}{1+\gamma\mu}\Psi_{k,t}^m \\
&+ \Big(\frac{\gamma^2\mu^2(\gamma-\eta)^2(\mu+L)}{(1+\gamma\mu)^2} + \frac{\gamma^4(\mu+L)^2L}{1+\gamma\mu}\Big)\sigma^2
\end{aligned} \tag{40}
$$

Since $\eta \leq \gamma$, we get $(\gamma-\eta)^2 \leq \gamma^2$. By using this fact, we obtain

$$
\begin{aligned}
\frac{\gamma\mu(\gamma-\eta)^2(\mu+L)}{1+\gamma\mu} + \gamma^2(\mu+L)(1-\gamma\mu) &\leq \frac{\gamma^3\mu(\mu+L)}{1+\gamma\mu} + \gamma^2(\mu+L)(1-\gamma\mu) \\
&= \gamma^2(\mu+L)(1-\gamma\mu+\frac{\gamma\mu}{1+\gamma\mu})
\end{aligned} \tag{41}
$$

It is easy to show that $1-\gamma\mu+\frac{\gamma\mu}{1+\gamma\mu} < 1$. Also, we get

$$
\frac{(1-\gamma\mu)^2}{1+\gamma\mu} < 1-\gamma\mu < 1-\gamma\mu+\frac{\gamma\mu}{1+\gamma\mu} \tag{42}
$$

From (40), (41), and (42) we finally get

$$\mathbb{E}[A^m_{k,t+1}|\mathcal{F}_{k,t}] \leq (1-\gamma\mu+\frac{\gamma\mu}{1+\gamma\mu})A^m_{k,t} + \Big(\frac{\gamma^2\mu^2(\gamma-\eta)^2(\mu+L)}{(1+\gamma\mu)^2} + \frac{\gamma^4(\mu+L)^2L}{1+\gamma\mu}\Big)\sigma^2$$

From this relationship between $A^m_{k,t+1}$ and $A^m_{k,t}$, we obtain the result of Proposition 14.

$$\begin{aligned}
&\mathbb{E}[A^m_{k,t}] \\
&\leq (1-\gamma\mu+\frac{\gamma\mu}{1+\gamma\mu})^t\mathbb{E}[A^m_{k,0}] + \Big(\frac{\gamma^2\mu^2(\gamma-\eta)^2(\mu+L)}{(1+\gamma\mu)^2} + \frac{\gamma^4(\mu+L)^2L}{1+\gamma\mu}\Big)\sigma^2 \cdot \frac{1-(1-\gamma\mu+\frac{\gamma\mu}{1+\gamma\mu})^t}{1-(1-\gamma\mu+\frac{\gamma\mu}{1+\gamma\mu})} \\
&= (1-\gamma\mu+\frac{\gamma\mu}{1+\gamma\mu})^t\mathbb{E}[A^m_{k,0}] + \Big(\frac{(\gamma-\eta)^2(\mu+L)}{1+\gamma\mu} + \frac{\gamma^2(\mu+L)^2L}{\mu^2}\Big)\sigma^2 \cdot \Big(1-(1-\gamma\mu+\frac{\gamma\mu}{1+\gamma\mu})^t\Big) \\
&\leq \mathbb{E}[A^m_{k,0}] + \Big(\frac{(\gamma-\eta)^2(\mu+L)}{1+\gamma\mu} + \frac{\gamma^2(\mu+L)^2L}{\mu^2}\Big) \cdot \Big(1-(1-\gamma\mu+\frac{\gamma\mu}{1+\gamma\mu})^t\Big)\sigma^2
\end{aligned}$$

Proposition 15: Let F be μ-strongly convex, and assume Assumption 2, 3, 4, then for $\alpha = \frac{1}{\gamma\mu}, \beta = \alpha+1, \gamma \in [\eta, \sqrt{\frac{\eta}{\mu}}], \eta, \gamma \in (0, \frac{1}{L}], \tau \geq 2$, FedAQ yields

$$\begin{aligned}
\frac{\mu}{2}\mathbb{E}[\|w^m_{k,\tau} - w_k\|^2] + \frac{L}{2}\mathbb{E}[\|w^{\text{ag},m}_{k,\tau} - w^{\text{ag}}_k\|^2] \leq &\Big(\frac{4\gamma^2\mu(\mu+L)}{(1+\gamma\mu)^2} + \frac{2L\gamma^2(\mu+L)}{1+\gamma\mu}\Big)\tau^2\mathbb{E}[\Psi_k] + (\gamma^2\mu+\eta^2L)\tau\sigma^2 \\
&+ \Big(\frac{(\gamma-\eta)^2\gamma^2\mu^2(\mu+L)}{(1+\gamma\mu)^2} + \frac{\gamma^4(\mu+L)^2L}{1+\gamma\mu}\Big)\frac{\tau^3\sigma^2}{2}
\end{aligned}$$

Proof of Proposition 15 Let's first bound $\mathbb{E}[\|w^m_{k,\tau} - w_k\|^2]$ and $\mathbb{E}[\|w^{\text{ag},m}_{k,\tau} - w^{\text{ag}}_k\|^2]$ individually.

$$\begin{aligned}
\mathbb{E}[\|w^m_{k,\tau} - w_k\|^2] &= \mathbb{E}[\|(w^m_{k,\tau} - w^m_{k,\tau-1}) + \cdots + (w^m_{k,1} - w^m_{k,0})\|^2] \\
&= \mathbb{E}\Big[\Big\|\sum_{t=0}^{\tau-1}\Big((1-\alpha^{-1})w^m_{k,t} + \alpha^{-1}w^{\text{md, m}}_{k,t} - w^m_{k,t} - \gamma g^m_{k,t}\Big)\Big\|^2\Big] \\
&= \mathbb{E}\Big[\Big\|\alpha^{-1}\sum_{t=0}^{\tau-1}(w^{\text{md},m}_{k,t} - w^m_{k,t}) - \gamma\sum_{t=0}^{\tau-1}g^m_{k,t}\Big\|^2\Big] \\
&\leq 2\alpha^{-2}\mathbb{E}[\|\sum_{t=0}^{\tau-1}(w^{\text{md},m}_{k,t} - w^m_{k,t})\|^2] + 2\gamma^2\mathbb{E}[\|\sum_{t=0}^{\tau-1}g^m_{k,t}\|^2] \\
&\leq 2\alpha^{-2}\tau\sum_{t=0}^{\tau-1}\mathbb{E}[\|w^{\text{md},m}_{k,t} - w^m_{k,t}\|^2] + 2\gamma^2\mathbb{E}[\|\sum_{t=0}^{\tau-1}\nabla F(w^{\text{md},m}_{k,t})\|^2] \\
&+ 2\gamma^2\mathbb{E}[\|\sum_{t=0}^{\tau-1}(g^m_{k,t} - \nabla F(w^{\text{md},m}_{k,t}))\|^2] \\
&\leq 2\alpha^{-2}(1-\beta^{-1})^2\tau\sum_{t=0}^{\tau-1}\mathbb{E}[\|w^m_{k,t} - w^{\text{ag},m}_{k,t}\|^2] + 2\gamma^2\tau\sum_{t=0}^{\tau-1}\mathbb{E}[\|\nabla F(w^{\text{md},m}_{k,t})\|^2] \\
&+ 2\gamma^2\sum_{t=0}^{\tau-1}\mathbb{E}[\|g^m_{k,t} - \nabla F(w^{\text{md},m}_{k,t})\|^2] \\
&= \tau\Big(\sum_{t=0}^{\tau-1}2\alpha^{-2}(1-\beta^{-1})^2\mathbb{E}[\|w^m_{k,t} - w^{\text{ag},m}_{k,t}\|^2] + 2\gamma^2\mathbb{E}[\|\nabla F(w^{\text{md},m}_{k,t})\|^2]\Big) + 2\tau\gamma^2\sigma^2
\end{aligned}$$

$$
\begin{aligned}
\mathbb{E}[\|w_{k,\tau}^{\mathrm{ag},m} - w_k^{\mathrm{ag}}\|^2] &= \mathbb{E}[\|\sum_{t=0}^{\tau-1}(w_{k,t+1}^{\mathrm{ag},m} - w_{k,t}^{\mathrm{ag},m})\|^2] \\
&= \mathbb{E}[\|\sum_{t=0}^{\tau-1}(w_{k,t}^{\mathrm{md},m} - w_{k,t}^{\mathrm{ag},m} - \eta g_{k,t}^m)\|^2] \\
&\leq 2\mathbb{E}[\|\sum_{t=0}^{\tau-1}(w_{k,t}^{\mathrm{md},m} - w_{k,t}^{\mathrm{ag},m})\|^2] + 2\eta^2\mathbb{E}[\|\sum_{t=0}^{\tau-1} g_{k,t}^m\|^2] \\
&= 2\beta^{-2}\mathbb{E}[\|\sum_{t=0}^{\tau-1}(w_{k,t}^m - w_{k,t}^{\mathrm{ag},m})\|^2] + 2\eta^2\mathbb{E}[\|\sum_{t=0}^{\tau-1}\nabla F(w_{k,t}^{\mathrm{md},m})\|^2] \\
&+ 2\eta^2\mathbb{E}[\|\sum_{t=0}^{\tau-1}(g_{k,t}^m - \nabla F(w_{k,t}^{\mathrm{md},m}))\|^2] \\
&\leq 2\beta^{-2}\tau\sum_{t=0}^{\tau-1}\mathbb{E}[\|w_{k,t}^m - w_{k,t}^{\mathrm{ag},m}\|^2] + 2\eta^2\tau\sum_{t=0}^{\tau-1}\mathbb{E}[\|\nabla F(w_{k,t}^{\mathrm{md},m})\|^2] \\
&+ 2\eta^2\sum_{t=0}^{\tau-1}\mathbb{E}[\|g_{k,t}^m - \nabla F(w_{k,t}^{\mathrm{md},m})\|^2] \\
&= \tau\Big(\sum_{t=0}^{\tau-1} 2\beta^{-2}\mathbb{E}[\|w_{k,t}^m - w_{k,t}^{\mathrm{ag},m}\|^2] + 2\eta^2\mathbb{E}[\|\nabla F(w_{k,t}^{\mathrm{md},m})\|^2]\Big) + 2\tau\eta^2\sigma^2
\end{aligned}
$$

Thus, by using the above results, we get

$$
\begin{aligned}
&\frac{\mu}{2}\mathbb{E}[\|w_{k,\tau}^m - w_k\|^2] + \frac{L}{2}\mathbb{E}[\|w_{k,\tau}^{\mathrm{ag},m} - w_k^{\mathrm{ag}}\|^2] \\
&\leq \tau\sum_{t=0}^{\tau-1}\Big\{\Big(\mu\alpha^{-2}(1-\beta^{-1})^2 + L\beta^{-2}\Big)\mathbb{E}[\|w_{k,t}^m - w_{k,t}^{\mathrm{ag},m}\|^2] + (\gamma^2\mu + \eta^2 L)\mathbb{E}[\|\nabla F(w_{k,t}^{\mathrm{md},m})\|^2]\Big\} \\
&+ (\gamma^2\mu + \eta^2 L)\tau\sigma^2 \\
&\leq \tau\sum_{t=0}^{\tau-1}\Big\{\Big(\mu\alpha^{-2}(1-\beta^{-1})^2 + L\beta^{-2}\Big)\mathbb{E}[\|w_{k,t}^m - w_{k,t}^{\mathrm{ag},m}\|^2] + (\gamma^2\mu + \eta^2 L)\frac{2L}{1+\gamma\mu}\mathbb{E}[\Psi_{k,t}^m]\Big\} \\
&+ (\gamma^2\mu + \eta^2 L)\tau\sigma^2 \ (\because (37)) \\
&\leq \tau\sum_{t=0}^{\tau-1}\Big\{\frac{\gamma^2\mu^2(\mu+L)}{(1+\gamma\mu)^2}\mathbb{E}[\|w_{k,t}^m - w_{k,t}^{\mathrm{ag},m}\|^2] + \gamma^2(\mu+L)\frac{2L}{1+\gamma\mu}\mathbb{E}[\Psi_{k,t}^m]\Big\} + (\gamma^2\mu + \eta^2 L)\tau\sigma^2 \\
&= \tau\Big(\sum_{t=0}^{\tau-1}\mathbb{E}[A_{k,t}^m]\Big) + (\gamma^2\mu + \eta^2 L)\tau\sigma^2
\end{aligned}
$$

By Proposition 14 and the fact $\Psi_{k,0}^m = \Psi_k$, we obtain

$$\begin{aligned}
&\frac{\mu}{2}\mathbb{E}[\|w_{k,\tau}^m - w_k\|^2] + \frac{L}{2}\mathbb{E}[\|w_{k,\tau}^{\mathrm{ag},m} - w_k^{\mathrm{ag}}\|^2] \\
&\leq \tau\Big\{\sum_{t=0}^{\tau-1}\mathbb{E}[A_{k,0}^m] + \Big(\frac{(\gamma-\eta)^2(\mu+L)}{1+\gamma\mu} + \frac{\gamma^2(\mu+L)^2L}{\mu^2}\Big)\cdot\Big(1-(1-\gamma\mu+\frac{\gamma\mu}{1+\gamma\mu})^t\Big)\sigma^2\Big\} \\
&+ (\gamma^2\mu+\eta^2L)\tau\sigma^2 \\
&= \tau^2\Big(\frac{\gamma^2\mu^2(\mu+L)}{(1+\gamma\mu)^2}\mathbb{E}[\|w_k - w_k^{\mathrm{ag}}\|^2] + \gamma^2(\mu+L)\frac{2L}{1+\gamma\mu}\mathbb{E}[\Psi_k]\Big) \\
&+ \tau\Big(\frac{(\gamma-\eta)^2(\mu+L)}{1+\gamma\mu} + \frac{\gamma^2(\mu+L)^2L}{\mu^2}\Big)\Big(\sum_{t=0}^{\tau-1}1-(1-\gamma\mu+\frac{\gamma\mu}{1+\gamma\mu})^t\Big)\sigma^2 + (\gamma^2\mu+\eta^2L)\tau\sigma^2
\end{aligned}$$

Before we get to the final result, let's find the upper bound for $\|w_k - w_k^{\mathrm{ag}}\|^2$, $\sum_{t=0}^{\tau-1}\Big(1-(1-\gamma\mu+\frac{\gamma\mu}{1+\gamma\mu})^t\Big)$

$$\begin{aligned}
\|w_k - w_k^{\mathrm{ag}}\|^2 &= \|w_k - w^* - (w_k^{\mathrm{ag}} - w^*)\|^2 \\
&\leq 2\|w_k - w^*\|^2 + 2\|w_k^{\mathrm{ag}} - w^*\|^2 \\
&\leq 2\|w_k - w^*\|^2 + 2\cdot\frac{2}{\mu}\Big(F(w_k^{\mathrm{ag}}) - F^* - \langle\nabla F(w^*), w_k^{\mathrm{ag}} - w^*\rangle\Big) \\
&= 2\|w_k - w^*\|^2 + \frac{4}{\mu}(F(w_k^{\mathrm{ag}}) - F^*) = \frac{4}{\mu}\Psi_k
\end{aligned}$$

$$\begin{aligned}
\sum_{t=0}^{\tau-1}\Big(1-(1-\gamma\mu+\frac{\gamma\mu}{1+\gamma\mu})^t\Big) &= \tau - \sum_{t=0}^{\tau-1}(1-\gamma\mu+\frac{\gamma\mu}{1+\gamma\mu})^t \\
&= \tau - \frac{1-(1-\gamma\mu+\frac{\gamma\mu}{1+\gamma\mu})^\tau}{1-(1-\gamma\mu+\frac{\gamma\mu}{1+\gamma\mu})} \\
&\leq \tau - \frac{1-(1-\frac{\gamma^2\mu^2}{1+\gamma\mu}\tau + (\frac{\gamma^2\mu^2}{1+\gamma\mu})^2\frac{\tau(\tau-1)}{2})}{\frac{\gamma^2\mu^2}{1+\gamma\mu}} \\
&= \frac{\gamma^2\mu^2}{1+\gamma\mu}\cdot\frac{\tau(\tau-1)}{2} \leq \frac{\gamma^2\mu^2}{1+\gamma\mu}\cdot\frac{\tau^2}{2}
\end{aligned}$$

Therefore, we conclude as below

$$\begin{aligned}
\frac{\mu}{2}\mathbb{E}[\|w_{k,\tau}^m - w_k\|^2] + \frac{L}{2}\mathbb{E}[\|w_{k,\tau}^{\mathrm{ag},m} - w_k^{\mathrm{ag}}\|^2] &\leq \Big(\frac{4\gamma^2\mu(\mu+L)}{(1+\gamma\mu)^2} + \frac{2L\gamma^2(\mu+L)}{1+\gamma\mu}\Big)\tau^2\mathbb{E}[\Psi_k] + (\gamma^2\mu+\eta^2L)\tau\sigma^2 \\
&+ \Big(\frac{(\gamma-\eta)^2\gamma^2\mu^2(\mu+L)}{(1+\gamma\mu)^2} + \frac{\gamma^4(\mu+L)^2L}{1+\gamma\mu}\Big)\frac{\tau^3\sigma^2}{2}
\end{aligned}$$

Proof of Lemma 10 By the definition of Ψ_k, $\Psi_{k,t}$ and Proposition 11,

$$\mathbb{E}[\Psi_{k+1}] = \mathbb{E}[\Psi_{k,\tau}] + \frac{\mu}{2}\mathbb{E}[\|w_{k+1} - \bar{w}_{k,\tau}\|^2] + \mathbb{E}[F(w_{k+1}^{\mathrm{ag}}) - \frac{1}{M}\sum_{m=1}^{M}F(w_{k,\tau}^{\mathrm{ag},m})]$$

Applying Proposition 12 and Proposition 13, we have

$$
\begin{aligned}
&\mathbb{E}[\Psi_{k+1}] \\
&\leq (1-\gamma\mu)^{\tau}\mathbb{E}[\Psi_k] + \frac{1}{2}(\eta^2 L + \frac{\gamma^2\mu}{M})\tau\sigma^2 \\
&+ \gamma\mu L\tau \cdot \max_{0\leq t<\tau} \mathbb{E}[\frac{1}{M}\sum_{m=1}^{M} \|\bar{w}_{k,t}^{\text{md}} - w_{k,t}^{\text{md},m}\| \| \frac{1}{1+\gamma\mu}(\bar{w}_{k,t} - w_{k,t}^m) + \frac{\gamma\mu}{1+\gamma\mu}(\bar{w}_{k,t}^{\text{ag}} - w_{k,t}^{\text{ag},m})\|] \\
&+ \frac{q\mu}{2M^2}\sum_{m=1}^{M}\mathbb{E}[\|w_{k,\tau}^m - w_k\|^2] + \frac{qL}{2M^2}\sum_{m=1}^{M}\mathbb{E}[\|w_{k,\tau}^{\text{ag},m} - w_k^{\text{ag}}\|^2] \\
&\leq (1-\gamma\mu)^{\tau}\mathbb{E}[\Psi_k] + \frac{1}{2}(\eta^2 L + \frac{\gamma^2\mu}{M})\tau\sigma^2 \\
&+ \gamma\mu L\tau \cdot \max_{0\leq t<\tau} \mathbb{E}[\frac{1}{M}\sum_{m=1}^{M} \|\bar{w}_{k,t}^{\text{md}} - w_{k,t}^{\text{md},m}\| \| \frac{1}{1+\gamma\mu}(\bar{w}_{k,t} - w_{k,t}^m) + \frac{\gamma\mu}{1+\gamma\mu}(\bar{w}_{k,t}^{\text{ag}} - w_{k,t}^{\text{ag},m})\|] \\
&+ \frac{q}{M}\Big[\Big(\frac{4\gamma^2\mu(\mu+L)}{(1+\gamma\mu)^2} + \frac{2L\gamma^2(\mu+L)}{1+\gamma\mu}\Big)\tau^2\mathbb{E}[\Psi_k] + (\gamma^2\mu + \eta^2 L)\tau\sigma^2 \\
&+ \Big(\frac{(\gamma-\eta)^2\gamma^2\mu^2(\mu+L)}{(1+\gamma\mu)^2} + \frac{\gamma^4(\mu+L)^2 L}{1+\gamma\mu}\Big)\frac{\tau^3\sigma^2}{2}\Big] \\
&= \Big\{(1-\gamma\mu)^{\tau} + \frac{q}{M}\Big(\frac{4\gamma^2\mu(\mu+L)}{(1+\gamma\mu)^2} + \frac{2L\gamma^2(\mu+L)}{1+\gamma\mu}\Big)\tau^2\Big\}\mathbb{E}[\Psi_k] + \frac{1}{2}(\eta^2 L + \frac{\gamma^2\mu}{M})\tau\sigma^2 \\
&+ \frac{q}{M}(\gamma^2\mu + \eta^2 L)\tau\sigma^2 + \frac{q}{2M}\Big(\frac{(\gamma-\eta)^2\gamma^2\mu^2(\mu+L)}{(1+\gamma\mu)^2} + \frac{\gamma^4(\mu+L)^2 L}{1+\gamma\mu}\Big)\tau^3\sigma^2 \\
&+ \gamma\mu L\tau \cdot \max_{0\leq t<\tau} \mathbb{E}[\frac{1}{M}\sum_{m=1}^{M} \|\bar{w}_{k,t}^{\text{md}} - w_{k,t}^{\text{md},m}\| \| \frac{1}{1+\gamma\mu}(\bar{w}_{k,t} - w_{k,t}^m) + \frac{\gamma\mu}{1+\gamma\mu}(\bar{w}_{k,t}^{\text{ag}} - w_{k,t}^{\text{ag},m})\|]
\end{aligned}
$$

The second inequality comes from Proposition 15. Then, let's define $C(\gamma,\tau)$ as

$$
C(\gamma,\tau) = (1-\gamma\mu)^{\tau} + \frac{q}{M}\Big(\frac{4\gamma^2\mu(\mu+L)}{(1+\gamma\mu)^2} + \frac{2L\gamma^2(\mu+L)}{1+\gamma\mu}\Big)\tau^2
$$

Finally, we obtain

$$
\begin{aligned}
\mathbb{E}[\Psi_{k+1}] \leq\ & C(\gamma,\tau)\mathbb{E}[\Psi_k] + \frac{1}{2}(\eta^2 L + \frac{\gamma^2\mu}{M})\tau\sigma^2 + \frac{q}{M}(\gamma^2\mu + \eta^2 L)\tau\sigma^2 \\
&+ \frac{q}{2M}\Big(\frac{(\gamma-\eta)^2\gamma^2\mu^2(\mu+L)}{(1+\gamma\mu)^2} + \frac{\gamma^4(\mu+L)^2 L}{1+\gamma\mu}\Big)\tau^3\sigma^2 + \gamma\mu L\tau \\
&\cdot \max_{0\leq t<\tau} \mathbb{E}[\frac{1}{M}\sum_{m=1}^{M} \|\bar{w}_{k,t}^{\text{md}} - w_{k,t}^{\text{md},m}\| \| \frac{1}{1+\gamma\mu}(\bar{w}_{k,t} - w_{k,t}^m) + \frac{\gamma\mu}{1+\gamma\mu}(\bar{w}_{k,t}^{\text{ag}} - w_{k,t}^{\text{ag},m})\|]
\end{aligned}
$$

5.3.2 Proof of Theorem 16

Theorem 16: Let F be μ-strongly convex, and assume Assumption 1, 2, 3, 4, then for $\alpha = \frac{1}{\gamma\mu}, \beta = \alpha + 1, \gamma = \max(\eta, \sqrt{\frac{\eta}{\mu\tau}}), \eta, \gamma \in (0, \frac{1}{L}], \tau \geq 2$, if the learning rate γ satisfies

$$
\Big(\mu^2 + \frac{q}{M}(\mu+L)(4\mu+2L)\Big)\gamma\tau \leq \frac{1}{2}\mu \tag{43}
$$

FedAQ yields

$$\mathbb{E}[\Psi_K] \leq \exp\Big(-\frac{1}{2}\max(\eta\mu, \sqrt{\frac{\eta\mu}{\tau}})K\tau\Big)\Psi_0 + (2q+1)(\frac{\eta^{\frac{1}{2}}\sigma^2}{\mu^{\frac{1}{2}}M\tau^{\frac{1}{2}}} + \frac{\eta\sigma^2}{M}) + 14\eta^2 L\tau\sigma^2$$
$$+ \frac{(780 + \frac{2q}{M})\eta^{\frac{3}{2}}L\tau^{\frac{1}{2}}\sigma^2}{\mu^{\frac{1}{2}}} + \frac{(\mu+L)(\mu^2+\mu L+L^2)q\eta^{\frac{3}{2}}\tau^{\frac{1}{2}}\sigma^2}{\mu^{\frac{5}{2}}M} + \frac{q\eta^3\tau^2(\mu+L)^2L\sigma^2}{\mu M}$$

Proof of Theorem 16 At first, due to the condition (43) in Theorem 16, we get

$$\begin{aligned}
C(\gamma,\tau) &= (1-\gamma\mu)^\tau + \frac{q}{M}\Big(\frac{4\gamma^2\mu(\mu+L)}{(1+\gamma\mu)^2} + \frac{2L\gamma^2(\mu+L)}{1+\gamma\mu}\Big)\tau^2 \\
&\leq 1 - \gamma\mu\tau + \gamma^2\mu^2\tau^2 + \frac{q}{M}\gamma^2(\mu+L)(4\mu+2L)\tau^2 \\
&= 1 - \gamma\mu\tau + \Big(\mu^2 + \frac{q}{M}(\mu+L)(4\mu+2L)\Big)\gamma^2\tau^2 \\
&\leq 1 - \frac{1}{2}\gamma\mu\tau \ (\because \text{ condition (43)})
\end{aligned}$$

The first inequality comes from the fact that $(1-\gamma\mu)^\tau \leq e^{-\gamma\mu\tau} \leq 1 - \gamma\mu\tau + \gamma^2\mu^2\tau^2$ when $0 \leq \gamma\mu \leq 1$. Also, it is trivial that $\gamma = \max(\eta, \sqrt{\frac{\eta}{\mu\tau}}) \in [\eta, \sqrt{\frac{\eta}{\mu}}]$. Thus, we can use Lemma 10. By using Lemma 10 and the above result, we obtain

$$\begin{aligned}
\mathbb{E}[\Psi_{k+1}] \leq (1 - \frac{1}{2}\gamma\mu\tau)\mathbb{E}[\Psi_k] + \frac{1}{2}(\eta^2 L + \frac{\gamma^2\mu}{M})\tau\sigma^2 \\
+ \frac{q}{M}(\gamma^2\mu + \eta^2 L)\tau\sigma^2 + \frac{q}{2M}\Big(\frac{(\gamma-\eta)^2\gamma^2\mu^2(\mu+L)}{(1+\gamma\mu)^2} + \frac{\gamma^4(\mu+L)^2L}{1+\gamma\mu}\Big)\tau^3\sigma^2 + \gamma\mu L\tau \\
\cdot \max_{0\leq t<\tau} \mathbb{E}[\frac{1}{M}\sum_{m=1}^{M} \|\bar{w}_{k,t}^{\text{md}} - w_{k,t}^{\text{md},m}\| \| \frac{1}{1+\gamma\mu}(\bar{w}_{k,t} - w_{k,t}^m) + \frac{\gamma\mu}{1+\gamma\mu}(\bar{w}_{k,t}^{\text{ag}} - w_{k,t}^{\text{ag},m})\|] \qquad (44)
\end{aligned}$$

By the Lemma B.3 in [43], we know that the below quantity is bounded.

$$\max_{0\leq t<\tau} \mathbb{E}[\frac{1}{M}\sum_{m=1}^{M} \|\bar{w}_{k,t}^{\text{md}} - w_{k,t}^{\text{md},m}\| \| \frac{1}{1+\gamma\mu}(\bar{w}_{k,t} - w_{k,t}^m) + \frac{\gamma\mu}{1+\gamma\mu}(\bar{w}_{k,t}^{\text{ag}} - w_{k,t}^{\text{ag},m})\|] \leq B$$
$$B = \begin{cases} 7\eta\gamma\tau\sigma^2\Big(1 + \frac{2\gamma^2\mu}{\eta}\Big)^{2\tau}, & \text{if } \gamma \in \Big(\eta, \sqrt{\frac{\eta}{\mu}}\Big] \\ 7\eta^2\tau\sigma^2, & \text{if } \gamma = \eta \end{cases}$$

Telescoping (44) yields

$$\begin{aligned}
\mathbb{E}[\Psi_K] &\leq (1 - \frac{1}{2}\gamma\mu\tau)^K\Psi_0 + \Big(\sum_{k'=0}^{K-1}(1 - \frac{1}{2}\gamma\mu\tau)^{k'}\Big) \cdot \Big[\frac{1}{2}(\eta^2 L + \frac{\gamma^2\mu}{M})\tau\sigma^2 + \gamma\mu L\tau B \\
&+ \frac{q}{M}(\gamma^2\mu + \eta^2 L)\tau\sigma^2 + \frac{q}{2M}\Big(\frac{(\gamma-\eta)^2\gamma^2\mu^2(\mu+L)}{(1+\gamma\mu)^2} + \frac{\gamma^4(\mu+L)^2L}{1+\gamma\mu}\Big)\tau^3\sigma^2\Big] \\
&\leq \exp\Big(-\frac{\gamma\mu\tau K}{2}\Big)\Psi_0 + \frac{\eta^2 L\sigma^2}{\gamma\mu} + \frac{\gamma\sigma^2}{M} + 2LB + 2q\Big(\frac{\gamma\sigma^2}{M} + \frac{\eta^2 L\sigma^2}{\gamma\mu M}\Big) \\
&+ \frac{q}{M}\Big(\frac{(\gamma-\eta)^2\gamma\mu(\mu+L)}{(1+\gamma\mu)^2} + \frac{\gamma^3(\mu+L)^2L}{(1+\gamma\mu)\mu}\Big)\tau^2\sigma^2
\end{aligned}$$

The last inequality comes from the fact that $\sum_{k'=0}^{K-1}(1-\frac{1}{2}\gamma\mu\tau)^{k'} \leq \frac{2}{\gamma\mu\tau}$. Since we plug in $\gamma = \max(\eta, \sqrt{\frac{\eta}{\mu\tau}})$, we can use Lemma B.4 in [43]. Therefore, we obtain

$$\begin{aligned}\mathbb{E}[\Psi_K] \leq \exp\Big(-\frac{1}{2}\max(\eta\mu, \sqrt{\frac{\eta\mu}{\tau}})K\tau\Big)\Psi_0 + \frac{\eta^{\frac{1}{2}}\sigma^2}{\mu^{\frac{1}{2}}M\tau^{\frac{1}{2}}} + \frac{\eta\sigma^2}{M} + \frac{780\eta^{\frac{3}{2}}L\tau^{\frac{1}{2}}\sigma^2}{\mu^{\frac{1}{2}}} + 14\eta^2 L\tau\sigma^2 \\ + \max\Big(\frac{2q\eta^{\frac{1}{2}}\sigma^2}{M\mu^{\frac{1}{2}}\tau^{\frac{1}{2}}}, \frac{2q\eta\sigma^2}{M}\Big) + \min\Big(\frac{2q\eta^{\frac{3}{2}}\tau^{\frac{1}{2}}L\sigma^2}{M\mu^{\frac{1}{2}}}, \frac{2q\eta L\sigma^2}{M\mu}\Big) \\ + \frac{q\tau^2\sigma^2}{M}\max\Big(\frac{\eta^{\frac{3}{2}}\mu(\mu+L)}{\mu^{\frac{3}{2}}\tau^{\frac{3}{2}}} + \frac{\eta^{\frac{3}{2}}(\mu+L)^2L}{\mu^{\frac{5}{2}}\tau^{\frac{3}{2}}}, \frac{\eta^3(\mu+L)^2L}{\mu}\Big)\end{aligned}$$

The first term stems directly from Lemma B.4 in [43]. Also, the last term comes from the fact that

$$\frac{(\gamma-\eta)^2\gamma\mu(\mu+L)}{(1+\gamma\mu)^2} + \frac{\gamma^3(\mu+L)^2L}{(1+\gamma\mu)\mu} \leq \begin{cases}\gamma^3\mu(\mu+L) + \frac{\gamma^3(\mu+L)^2L}{\mu}, & \text{if } \gamma \neq \eta \\ \frac{\eta^3(\mu+L)^2L}{\mu}, & \text{if } \gamma = \eta\end{cases}$$

Therefore, by simple inequalities such as $\max(a,b) \leq a+b$ and $\min(a,b) \leq a$, we ultimately get

$$\begin{aligned}\mathbb{E}[\Psi_K] \leq \exp\Big(-\frac{1}{2}\max(\eta\mu, \sqrt{\frac{\eta\mu}{\tau}})K\tau\Big)\Psi_0 + \frac{(2q+1)\eta^{\frac{1}{2}}\sigma^2}{\mu^{\frac{1}{2}}M\tau^{\frac{1}{2}}} + \frac{(2q+1)\eta\sigma^2}{M} + 14\eta^2L\tau\sigma^2 \\ + \frac{(780+\frac{2q}{M})\eta^{\frac{3}{2}}L\tau^{\frac{1}{2}}\sigma^2}{\mu^{\frac{1}{2}}} + \frac{(\mu+L)(\mu^2+\mu L+L^2)q\eta^{\frac{3}{2}}\tau^{\frac{1}{2}}\sigma^2}{\mu^{\frac{5}{2}}M} + \frac{q\eta^3\tau^2(\mu+L)^2L\sigma^2}{\mu M} \qquad (45)\end{aligned}$$

5.3.3 Proof of Corollary 17

Corollary 17: Let $C_1, C_2,$ and η_0 as below. Note that $T = K\tau$.

$$\begin{aligned}C_1 = \frac{(\mu+L)(\mu^2+\mu L+L^2)q}{\mu^{\frac{5}{2}}},\ C_2 = \frac{q(\mu+L)^2L}{\mu} \\ \eta_0 = \frac{4\tau}{\mu T^2}\log^2\Big(e + \min(\frac{\mu MT\Psi_0}{(2q+1)\sigma^2}, \frac{\mu^2T^3\Psi_0}{L\tau^2\sigma^2}, \frac{\mu^3MT^3\Psi_0}{(\mu^{\frac{3}{2}}C_1+8C_2)\tau^2\sigma^2})\Big)\end{aligned}$$

Then for $\eta = \min(\frac{1}{L}, \eta_0)$, FedAQ yields

$$\begin{aligned}\mathbb{E}[\Psi_K] \leq \min\Big(\exp(-\frac{\mu T}{2L}), \exp(-\frac{\mu^{\frac{1}{2}}T}{2L^{\frac{1}{2}}\tau^{\frac{1}{2}}})\Big)\Psi_0 \\ + \frac{7(2q+1)\sigma^2}{\mu MT}\log^2\Big(e + \frac{\mu MT\Psi_0}{(2q+1)\sigma^2}\Big) \qquad (46) \\ + \frac{(6465+\frac{16q}{M})L\tau^2\sigma^2}{\mu^2T^3}\log^4\Big(e + \frac{\mu^2T^3\Psi_0}{L\tau^2\sigma^2}\Big) \qquad (47) \\ + \frac{9(\mu^{\frac{3}{2}}C_1+8C_2)\tau^2\sigma^2}{\mu^3MT^3}\log^6\Big(e + \frac{\mu^3MT^3\Psi_0}{(\mu^{\frac{3}{2}}C_1+8C_2)\tau^2\sigma^2}\Big) \qquad (48)\end{aligned}$$

Proof of Corollary 17 Let's decompose the final result (45) of the Theorem 16 into a decreasing term and an increasing term. We denote the decreasing term ψ_1 and the increasing term ψ_2 as below.

$$\begin{aligned}
\psi_1(\eta) &= \exp\Big(-\frac{1}{2}\max(\eta\mu, \sqrt{\frac{\eta\mu}{\tau}})T\Big)\Psi_0 \\
\psi_2(\eta) &= \frac{(2q+1)\eta^{\frac{1}{2}}\sigma^2}{\mu^{\frac{1}{2}}M\tau^{\frac{1}{2}}} + \frac{(2q+1)\eta\sigma^2}{M} + \frac{(780+\frac{2q}{M})\eta^{\frac{3}{2}}L\tau^{\frac{1}{2}}\sigma^2}{\mu^{\frac{1}{2}}} + 14\eta^2 L\tau\sigma^2 \\
&\quad + \frac{(\mu+L)(\mu^2+\mu L+L^2)q\eta^{\frac{3}{2}}\tau^{\frac{1}{2}}\sigma^2}{\mu^{\frac{5}{2}}M} + \frac{q\eta^3\tau^2(\mu+L)^2L\sigma^2}{\mu M}
\end{aligned}$$

Since ψ_1 is the decreasing term, we have

$$\psi_1(\eta) \le \psi_1(\frac{1}{L}) + \psi_1(\eta_0) \tag{49}$$

where

$$\begin{aligned}
\psi_1(\frac{1}{L}) &= \min\Big(\exp(-\frac{\mu T}{2L}), \exp(-\frac{\mu^{\frac{1}{2}}T}{2L^{\frac{1}{2}}\tau^{\frac{1}{2}}})\Big)\Psi_0 \\
\psi_1(\eta_0) &\le \exp\Big(-\frac{1}{2}\sqrt{\frac{\eta_0\mu}{\tau}}T\Big) \\
&= \Big(e + \min(\frac{\mu MT\Psi_0}{(2q+1)\sigma^2}, \frac{\mu^2T^3\Psi_0}{L\tau^2\sigma^2}, \frac{\mu^3MT^3\Psi_0}{(\mu^{\frac{3}{2}}C_1+8C_2)\tau^2\sigma^2})\Big)^{-1}\Psi_0 \\
&\le \frac{(2q+1)\sigma^2}{\mu MT} + \frac{L\tau^2\sigma^2}{\mu^2T^3} + \frac{(\mu^{\frac{3}{2}}C_1+8C_2)\tau^2\sigma^2}{\mu^3MT^3}
\end{aligned}$$

Since ψ_2 is the increasing term, we have

$$\begin{aligned}
&\psi_2(\eta) \\
&\le \psi_2(\eta_0) \\
&\le \frac{2(2q+1)\sigma^2}{\mu MT}\log\Big(e + \frac{\mu MT\Psi_0}{(2q+1)\sigma^2}\Big) + \frac{4(2q+1)\tau\sigma^2}{\mu MT^2}\log^2\Big(e + \frac{\mu MT\Psi_0}{(2q+1)\sigma^2}\Big) \\
&+ \frac{8(780+\frac{2q}{M})L\tau^2\sigma^2}{\mu^2T^3}\log^3\Big(e + \frac{\mu^2T^3\Psi_0}{L\tau^2\sigma^2}\Big) + \frac{224L\tau^3\sigma^2}{\mu^2T^4}\log^4\Big(e + \frac{\mu^2T^3\Psi_0}{L\tau^2\sigma^2}\Big) \\
&+ \frac{8C_1\tau^2\sigma^2}{\mu^{\frac{3}{2}}MT^3}\log^3\Big(e + \frac{\mu^3MT^3\Psi_0}{(\mu^{\frac{3}{2}}C_1+8C_2)\tau^2\sigma^2}\Big) + \frac{64C_2\tau^5\sigma^2}{\mu^3MT^6}\log^6\Big(e + \frac{\mu^3MT^3\Psi_0}{(\mu^{\frac{3}{2}}C_1+8C_2)\tau^2\sigma^2}\Big) \\
&\le \frac{6(2q+1)\sigma^2}{\mu MT}\log^2\Big(e + \frac{\mu MT\Psi_0}{(2q+1)\sigma^2}\Big) + \frac{(6464+\frac{16q}{M})L\tau^2\sigma^2}{\mu^2T^3}\log^4\Big(e + \frac{\mu^2T^3\Psi_0}{L\tau^2\sigma^2}\Big) \\
&+ \frac{8(\mu^{\frac{3}{2}}C_1+8C_2)\tau^2\sigma^2}{\mu^3MT^3}\log^6\Big(e + \frac{\mu^3MT^3\Psi_0}{(\mu^{\frac{3}{2}}C_1+8C_2)\tau^2\sigma^2}\Big)
\end{aligned} \tag{50}$$

The last inequality comes from $\frac{\tau}{T} \leq 1$. Therefore, by combining (49) and (50), we finally get

$$\begin{aligned}
\mathbb{E}[\Psi_K] &\leq \psi_1(\eta) + \psi_2(\eta) \\
&\leq \psi_1(\frac{1}{L}) + \psi_1(\eta_0) + \psi_2(\eta_0) \\
&\leq \min\Big(\exp(-\frac{\mu T}{2L}), \exp(-\frac{\mu^{\frac{1}{2}} T}{2L^{\frac{1}{2}}\tau^{\frac{1}{2}}})\Big)\Psi_0 + \frac{7(2q+1)\sigma^2}{\mu MT}\log^2\Big(e + \frac{\mu MT\Psi_0}{(2q+1)\sigma^2}\Big) \\
&+ \frac{(6465 + \frac{16q}{M})L\tau^2\sigma^2}{\mu^2 T^3}\log^4\Big(e + \frac{\mu^2 T^3\Psi_0}{L\tau^2\sigma^2}\Big) \\
&+ \frac{9(\mu^{\frac{3}{2}}C_1 + 8C_2)\tau^2\sigma^2}{\mu^3 MT^3}\log^6\Big(e + \frac{\mu^3 MT^3\Psi_0}{(\mu^{\frac{3}{2}}C_1 + 8C_2)\tau^2\sigma^2}\Big)
\end{aligned}$$

5.3.4 Why the Condition (43) is Satisfied

The synchronization rounds K required for linear speedup in M for FedAQ is $\tilde{\mathcal{O}}((\frac{M}{1+q})^{\frac{1}{2}})$ (See Remark 8). Since we derive this result from Theorem 16, we should show that $K = \tilde{\mathcal{O}}((\frac{M}{1+q})^{\frac{1}{2}})$ satisfies the condition (43) in Theorem 16.

$$\Big(\mu^2 + \frac{q}{M}(\mu + L)(4\mu + 2L)\Big)\gamma\tau \leq \frac{1}{2}\mu$$

We rewrite the above condition as below.

$$\gamma\tau \leq \frac{\mu}{2\mu^2 + \frac{2q}{M}(\mu + L)(4\mu + 2L)} \tag{51}$$

We know $\gamma = \max(\eta, \sqrt{\frac{\eta}{\mu\tau}})$ and $\eta = \min(\frac{1}{L}, \eta_0)$. Since η_0 becomes smaller and smaller as T increases, we assume $\eta = \eta_0$ here. Therefore, we get

$$\begin{aligned}
\gamma\tau &= \max(\eta_0\tau, \sqrt{\frac{\eta_0\tau}{\mu}}) \\
&= \max\Big(\frac{4\tau^2}{\mu T^2}\log^2\Big(e + \min(\frac{\mu MT\Psi_0}{(2q+1)\sigma^2}, \frac{\mu^2 T^3\Psi_0}{L\tau^2\sigma^2}, \frac{\mu^3 MT^3\Psi_0}{(\mu^{\frac{3}{2}}C_1 + 8C_2)\tau^2\sigma^2})\Big), \\
&\frac{2\tau}{\mu T}\log\Big(e + \min(\frac{\mu MT\Psi_0}{(2q+1)\sigma^2}, \frac{\mu^2 T^3\Psi_0}{L\tau^2\sigma^2}, \frac{\mu^3 MT^3\Psi_0}{(\mu^{\frac{3}{2}}C_1 + 8C_2)\tau^2\sigma^2})\Big)\Big)
\end{aligned}$$

Note that $K = \frac{T}{\tau} = \tilde{\mathcal{O}}((\frac{M}{1+q})^{\frac{1}{2}}) = C(\frac{M}{1+q})^{\frac{1}{2}}\log(T)$ because $\tilde{\mathcal{O}}$ contains hidden multiplicative polylog factors with respect to T. We can assume T is sufficiently large here. Then, we have

$$\begin{aligned}
\gamma\tau &= \max\Big(\frac{4(1+q)}{\mu C^2 M\log^2(T)}\log^2\Big(e + \min(\frac{\mu MT\Psi_0}{(2q+1)\sigma^2}, \frac{\mu^2 T^3\Psi_0}{L\tau^2\sigma^2}, \frac{\mu^3 MT^3\Psi_0}{(\mu^{\frac{3}{2}}C_1 + 8C_2)\tau^2\sigma^2})\Big), \\
&\frac{2(1+q)^{\frac{1}{2}}}{\mu CM^{\frac{1}{2}}\log(T)}\log\Big(e + \min(\frac{\mu MT\Psi_0}{(2q+1)\sigma^2}, \frac{\mu^2 T^3\Psi_0}{L\tau^2\sigma^2}, \frac{\mu^3 MT^3\Psi_0}{(\mu^{\frac{3}{2}}C_1 + 8C_2)\tau^2\sigma^2})\Big)\Big) \\
&\leq \max\Big(\frac{4(1+q)}{\mu C^2 M\log^2(T)}\log^2\Big(\frac{2\mu MT\Psi_0}{(2q+1)\sigma^2}\Big), \frac{2(1+q)^{\frac{1}{2}}}{\mu CM^{\frac{1}{2}}\log(T)}\log\Big(\frac{2\mu MT\Psi_0}{(2q+1)\sigma^2}\Big)\Big)
\end{aligned}$$

For an arbitrary constant $k_1 > 0$, it is easy to show that $\lim_{T\to\infty} \frac{\log(k_1 T)}{\log(T)} = 1$. Thus, we obtain

$$\begin{aligned}\gamma\tau &\leq \max\Big(\frac{4(1+q)}{\mu C^2 M \log^2(T)} \log^2\Big(\frac{2\mu M T \Psi_0}{(2q+1)\sigma^2}\Big), \frac{2(1+q)^{\frac{1}{2}}}{\mu C M^{\frac{1}{2}} \log(T)} \log\Big(\frac{2\mu M T \Psi_0}{(2q+1)\sigma^2}\Big)\Big) \\ &\simeq \max\Big(\frac{4(1+q)}{\mu C^2 M}, \frac{2(1+q)^{\frac{1}{2}}}{\mu C M^{\frac{1}{2}}}\Big) \\ &\leq \frac{\mu}{2\mu^2 + \frac{2q}{M}(\mu+L)(4\mu+2L)}\end{aligned}$$

Finally, we conclude that there exists a constant C that meets the last inequality. Therefore, $K = \tilde{\mathcal{O}}((\frac{M}{1+q})^{\frac{1}{2}})$ satisfies the condition (43).

5.4 Proof Details for FedAQ under Condition Set (34)

We use notations defined in section 5.3 here as well. We newly define $\Phi^m_{k,t}, \Phi_{k,t}, \Phi_k, B^m_{k,t}$ as below.

$$\begin{aligned}\Phi^m_{k,t} &= F(w^{\text{ag},m}_{k,t}) - F^* + \frac{1}{6}\mu\|w^m_{k,t} - w^*\|^2 \\ \Phi_{k,t} &= F(\bar{w}^{\text{ag}}_{k,t}) - F^* + \frac{1}{6}\mu\|\bar{w}_{k,t} - w^*\|^2 \\ \Phi_k &:= \Phi_{k,0} = F(w^{\text{ag}}_k) - F^* + \frac{1}{6}\mu\|w_k - w^*\|^2 \\ B^m_{k,t} &= \Big(\frac{\mu\alpha^{-2}}{3}(1-\beta^{-1})^2 + L\beta^{-2}\Big)\|w^m_{k,t} - w^{\text{ag},m}_{k,t}\|^2 + \gamma^2(\frac{\mu}{3}+L)\frac{2\alpha^2-\alpha}{2\alpha^2-1}\cdot 2L\Phi^m_{k,t}\end{aligned}$$

The flow of proof is similar to section 5.3. We need one more condition $\gamma\mu \leq \frac{3}{4}$ to show the convergence of FedAQ under the parameter condition set (34).

5.4.1 Proof of Lemma 5

In order to prove Lemma 5, we first introduce five crucial Propositions for proving Lemma 5. Then, we prove Lemma 5 by using Propositions in the last part of this section.

Proposition 18: Let Assumption 1 hold and consider any k synchronization round. Then, we can decompose the expectation as follows:

$$\begin{aligned}\mathbb{E}[\|w_{k+1} - w^*\|^2] &= \mathbb{E}[\|w_{k+1} - \bar{w}_{k,\tau}\|^2] + \mathbb{E}[\|\bar{w}_{k,\tau} - w^*\|^2] \\ \mathbb{E}[F(w^{\text{ag}}_{k+1}) - F^*] &= \mathbb{E}[F(w^{\text{ag}}_{k+1}) - F(\bar{w}^{\text{ag}}_{k,\tau})] + \mathbb{E}[F(\bar{w}^{\text{ag}}_{k,\tau}) - F^*]\end{aligned}$$

Proof of Proposition 18 The second equality is trivial. The first equality is the same as one in Proposition 11.

Proposition 19: Let F be μ-strongly convex, and assume Assumption 2, 3, 4, then for $\alpha = \frac{3}{2\gamma\mu} - \frac{1}{2}, \beta = \frac{2\alpha^2-1}{\alpha-1}, \gamma \in [\eta, \sqrt{\frac{\eta}{\mu}}], \eta \in (0, \frac{1}{L}]$, FedAQ yields

$$\begin{aligned}\mathbb{E}[\Phi_{k,\tau}] \leq (1 - \frac{1}{3}\gamma\mu)^\tau \mathbb{E}[\Phi_k] + (\frac{\eta^2 L}{2} + \frac{\gamma^2\mu}{6})\frac{\tau\sigma^2}{M} \\ + \gamma\tau \cdot \max_{0\leq t<\tau} \mathbb{E}[\|\nabla F(\bar{w}^{\text{md}}_{k,t}) - \frac{1}{M}\sum_{m=1}^M \nabla F(w^{\text{md},m}_{k,t})\|^2]\end{aligned}$$

Proof of Proposition 19 We refer to the proof of Lemma C.2 in [43]. There is no quantization between $\Phi_{k,\tau}$ and Φ_k. Thus, we can directly apply useful inequalities in the proof of Lemma C.2 in [43] to our proof. Then, we obtain

$$\mathbb{E}[\Phi_{k,t+1}|\mathcal{F}_{k,t}] \leq (1-\frac{1}{3}\gamma\mu)\Phi_{k,t} + (\frac{\eta^2 L}{2} + \frac{\gamma^2\mu}{6})\frac{\sigma^2}{M} + \gamma\|\nabla F(\bar{w}_{k,t}^{\mathrm{md}}) - \frac{1}{M}\sum_{m=1}^{M}\nabla F(w_{k,t}^{\mathrm{md},m})\|^2$$

From the above relationship between $\Phi_{k,t+1}$ and $\Phi_{k,t}$, we get

$$\begin{aligned}
\mathbb{E}[\Phi_{k,\tau}] &\leq (1-\frac{1}{3}\gamma\mu)^\tau\mathbb{E}[\Phi_k] + \Big(\sum_{t=0}^{\tau-1}(1-\frac{1}{3}\gamma\mu)^t\Big)\cdot(\frac{\eta^2 L}{2} + \frac{\gamma^2\mu}{6})\frac{\sigma^2}{M} \\
&\quad + \gamma\sum_{t=0}^{\tau-1}\Big\{(1-\frac{1}{3}\gamma\mu)^{\tau-t-1}\mathbb{E}[\|\nabla F(\bar{w}_{k,t}^{\mathrm{md}}) - \frac{1}{M}\sum_{m=1}^{M}\nabla F(w_{k,t}^{\mathrm{md},m})\|^2]\Big\} \\
&\leq (1-\frac{1}{3}\gamma\mu)^\tau\mathbb{E}[\Phi_k] + (\frac{\eta^2 L}{2} + \frac{\gamma^2\mu}{6})\frac{\tau\sigma^2}{M} \\
&\quad + \gamma\tau\cdot\max_{0\leq t<\tau}\mathbb{E}[\|\nabla F(\bar{w}_{k,t}^{\mathrm{md}}) - \frac{1}{M}\sum_{m=1}^{M}\nabla F(w_{k,t}^{\mathrm{md},m})\|^2]
\end{aligned}$$

Proposition 20: Let Assumption 1 hold. Then, we have

$$\begin{aligned}
\mathbb{E}[\|w_{k+1} - \bar{w}_{k,\tau}\|^2] &\leq \frac{q}{M^2}\sum_{m=1}^{M}\mathbb{E}[\|w_{k,\tau}^m - w_k\|^2] \\
\mathbb{E}[F(w_{k+1}^{\mathrm{ag}}) - F(\bar{w}_{k,\tau}^{\mathrm{ag}})] &\leq \frac{qL}{2M^2}\sum_{m=1}^{M}\mathbb{E}[\|w_{k,\tau}^{\mathrm{ag},m} - w_k^{\mathrm{ag}}\|^2]
\end{aligned}$$

Proof of Proposition 20 The first inequality is the same as one in Proposition 13. The proof of the second inequality is similar to Proposition 13 as well.

$$\begin{aligned}
\mathbb{E}[F(w_{k+1}^{\mathrm{ag}}) - F(\bar{w}_{k,\tau}^{\mathrm{ag}})] &= \mathbb{E}[F(w_k^{\mathrm{ag}} + \frac{1}{M}\sum_{m=1}^{M}Q(w_{k,\tau}^{\mathrm{ag},m} - w_k^{\mathrm{ag}})) - F(\frac{1}{M}\sum_{m=1}^{M}w_{k,\tau}^{\mathrm{ag},m})] \\
&\leq \mathbb{E}\Big[\langle\nabla F(\frac{1}{M}\sum_{m=1}^{M}w_{k,\tau}^{\mathrm{ag},m}), \frac{1}{M}\sum_{m=1}^{M}\Big(Q(w_{k,\tau}^{\mathrm{ag},m} - w_k^{\mathrm{ag}}) \\
&\quad - (w_{k,\tau}^{\mathrm{ag},m} - w_k^{\mathrm{ag}})\Big)\rangle + \frac{L}{2}\|\frac{1}{M}\sum_{m=1}^{M}Q(w_{k,\tau}^{\mathrm{ag},m} - w_k^{\mathrm{ag}}) - (w_{k,\tau}^{\mathrm{ag},m} - w_k^{\mathrm{ag}})\|^2\Big] \\
&= \frac{L}{2}\mathbb{E}[\|\frac{1}{M}\sum_{m=1}^{M}Q(w_{k,\tau}^{\mathrm{ag},m} - w_k^{\mathrm{ag}}) - (w_{k,\tau}^{\mathrm{ag},m} - w_k^{\mathrm{ag}})\|^2] \\
&= \frac{L}{2M^2}\sum_{m=1}^{M}\mathbb{E}[\|Q(w_{k,\tau}^{\mathrm{ag},m} - w_k^{\mathrm{ag}}) - (w_{k,\tau}^{\mathrm{ag},m} - w_k^{\mathrm{ag}})\|^2] \\
&\leq \frac{qL}{2M^2}\sum_{m=1}^{M}\mathbb{E}[\|w_{k,\tau}^{\mathrm{ag},m} - w_k^{\mathrm{ag}}\|^2]
\end{aligned}$$

Proposition 21: Let F be μ-strongly convex, and assume Assumption 2, 3, 4, then for $\alpha = \frac{3}{2\gamma\mu} - \frac{1}{2}, \beta = \frac{2\alpha^2-1}{\alpha-1}, \gamma \in [\eta, \sqrt{\frac{\eta}{\mu}}], \eta, \gamma \in (0, \frac{1}{L}], \gamma\mu \leq \frac{3}{4}$, we get

$$\mathbb{E}[B_{k,t}^m] \leq \mathbb{E}[B_{k,0}^m] + \left(\left(\frac{\mu}{3} (\frac{2\alpha-1}{2\alpha^2-1})^2 + L(\frac{\alpha-1}{2\alpha^2-1})^2 \right) \cdot (\gamma-\eta)^2 + \gamma^4 (\frac{\mu}{3}+L)^2 \frac{2\alpha^2-\alpha}{2\alpha^2-1} L \right)$$
$$\cdot \frac{1+\frac{1}{2}\alpha^{-1}}{\frac{1}{4}\alpha^{-2}} \cdot \left(1 - (1 - \frac{1}{2}\alpha^{-1} + \frac{\frac{1}{2}\alpha^{-1}}{1+\frac{1}{2}\alpha^{-1}})^t \right) \sigma^2$$

Proof of Proposition 21 From the notation mentioned in the beginning of section 5.4,

$$\mathbb{E}[B_{k,t+1}^m | \mathcal{F}_{k,t}] = \left(\frac{\mu\alpha^{-2}}{3}(1-\beta^{-1})^2 + L\beta^{-2} \right) \mathbb{E}[\|w_{k,t+1}^m - w_{k,t+1}^{\mathrm{ag},m}\|^2 | \mathcal{F}_{k,t}]$$
$$+ \gamma^2 (\frac{\mu}{3} + L) \frac{2\alpha^2-\alpha}{2\alpha^2-1} \cdot 2L\mathbb{E}[\Phi_{k,t+1}^m | \mathcal{F}_{k,t}] \tag{52}$$

Thus, let's sequentially compute $\mathbb{E}[\|w_{k,t+1}^m - w_{k,t+1}^{\mathrm{ag},m}\|^2 | \mathcal{F}_{k,t}]$ and $\mathbb{E}[\Phi_{k,t+1}^m | \mathcal{F}_{k,t}]$.

$$\begin{aligned}
\mathbb{E}[\|w_{k,t+1}^m - w_{k,t+1}^{\mathrm{ag},m}\|^2 | \mathcal{F}_{k,t}] &= \mathbb{E}[\|(1-\alpha^{-1})w_{k,t}^m + \alpha^{-1} w_{k,t}^{\mathrm{md},m} - \gamma g_{k,t}^m - w_{k,t}^{\mathrm{md},m} + \eta g_{k,t}^m\|^2 | \mathcal{F}_{k,t}] \\
&= \mathbb{E}[\|(1-\alpha^{-1})(w_{k,t}^m - w_{k,t}^{\mathrm{md},m}) - (\gamma-\eta) g_{k,t}^m\|^2 | \mathcal{F}_{k,t}] \ (\leftarrow \gamma \geq \eta) \\
&= \|(1-\alpha^{-1})(w_{k,t}^m - w_{k,t}^{\mathrm{md},m}) - (\gamma-\eta)\nabla F(w_{k,t}^{\mathrm{md},m})\|^2 \\
&+ (\gamma-\eta)^2 \mathbb{E}[\|\nabla F(w_{k,t}^{\mathrm{md},m}) - g_{k,t}^m\|^2 | \mathcal{F}_{k,t}] \\
&\leq (1-\alpha^{-1})^2 \|w_{k,t}^m - w_{k,t}^{\mathrm{md},m}\|^2 + (\gamma-\eta)^2 \|\nabla F(w_{k,t}^{\mathrm{md},m})\|^2 \\
&+ (\gamma-\eta)^2\sigma^2 - 2(\gamma-\eta)\langle (1-\alpha^{-1})(w_{k,t}^m - w_{k,t}^{\mathrm{md},m}), \nabla F(w_{k,t}^{\mathrm{md},m}) \rangle \\
&\leq (1-\alpha^{-1})^2(1+2\alpha^{-1}) \|w_{k,t}^m - w_{k,t}^{\mathrm{md},m}\|^2 \\
&+ (\gamma-\eta)^2 (1+\frac{\alpha}{2}) \|\nabla F(w_{k,t}^{\mathrm{md},m})\|^2 + (\gamma-\eta)^2\sigma^2
\end{aligned}$$

Here, we need to bound $\|\nabla F(w_{k,t}^{\mathrm{md},m})\|^2$.

$$\begin{aligned}
\|\nabla F(w_{k,t}^{\mathrm{md},m})\|^2 &\leq 2L(F(w_{k,t}^{\mathrm{md},m}) - F^*) \ (\because \text{ Assumption 3}) \\
&\leq 2L\left(\beta^{-1}(F(w_{k,t}^m) - F(w^*)) + (1-\beta^{-1})(F(w_{k,t}^{\mathrm{ag},m}) - F^*) \right) \\
&\leq \beta^{-1} L^2 \|w_{k,t}^m - w^*\|^2 + 2(1-\beta^{-1}) L (F(w_{k,t}^{\mathrm{ag},m}) - F^*) \\
&= \frac{\alpha-1}{2\alpha^2-1} L^2 \|w_{k,t}^m - w^*\|^2 + 2L \cdot \frac{2\alpha^2-\alpha}{2\alpha^2-1} (F(w_{k,t}^{\mathrm{ag},m}) - F^*) \\
&\leq \frac{\frac{\mu}{3}(2\alpha^2-\alpha)}{2\alpha^2-1} L \|w_{k,t}^m - w^*\|^2 + 2L \cdot \frac{2\alpha^2-\alpha}{2\alpha^2-1} (F(w_{k,t}^{\mathrm{ag},m}) - F^*) \\
&= \frac{2\alpha^2-\alpha}{2\alpha^2-1} \cdot 2L\Phi_{k,t}^m
\end{aligned} \tag{53}$$

It is easy to show $(\alpha-1)L \leq \frac{\mu}{3}(2\alpha^2-\alpha)$ by using the fact $\gamma L \leq 1$. Therefore, we finally get

$$\begin{aligned}
&\mathbb{E}[\|w_{k,t+1}^m - w_{k,t+1}^{\mathrm{ag},m}\|^2 | \mathcal{F}_{k,t}] \\
&\leq (1-\alpha^{-1})^2(1+2\alpha^{-1}) \|w_{k,t}^m - w_{k,t}^{\mathrm{md},m}\|^2 + (\gamma-\eta)^2(1+\frac{\alpha}{2}) \|\nabla F(w_{k,t}^{\mathrm{md},m})\|^2 + (\gamma-\eta)^2\sigma^2 \\
&\leq (1-\alpha^{-1})^2(1+2\alpha^{-1}) \|w_{k,t}^m - w_{k,t}^{\mathrm{md},m}\|^2 + (\gamma-\eta)^2(1+\frac{\alpha}{2}) (\frac{2\alpha^2-\alpha}{2\alpha^2-1} \cdot 2L\Phi_{k,t}^m) \\
&+ (\gamma-\eta)^2\sigma^2
\end{aligned} \tag{54}$$

Now, let's compute $\mathbb{E}[\Phi_{k,t+1}^m|\mathcal{F}_{k,t}]$. We need to compute $\mathbb{E}[\|w_{k,t+1}^m - w^*\|^2|\mathcal{F}_{k,t}]$ and $\mathbb{E}[F(w_{k,t+1}^{\text{ag},m}) - F^*|\mathcal{F}_{k,t}]$ first.

$$
\begin{aligned}
&\mathbb{E}[\|w_{k,t+1}^m - w^*\|^2|\mathcal{F}_{k,t}] \\
&= \mathbb{E}[\|(1-\alpha^{-1})w_{k,t}^m + \alpha^{-1}w_{k,t}^{\text{md},m} - \gamma g_{k,t}^m - w^*\|^2|\mathcal{F}_{k,t}] \\
&\le \|(1-\alpha^{-1})w_{k,t}^m + \alpha^{-1}w_{k,t}^{\text{md},m} - \gamma\nabla F(w_{k,t}^{\text{md},m}) - w^*\|^2 + \gamma^2\sigma^2 \\
&\le (1+\frac{1}{2}\alpha^{-1})\|(1-\alpha^{-1})w_{k,t}^m + \alpha^{-1}w_{k,t}^{\text{md},m} - \gamma\nabla F(w_{k,t}^{\text{md},m}) - w^*\|^2 + \gamma^2\sigma^2 \\
&= (1+\frac{1}{2}\alpha^{-1})\|(1-\alpha^{-1})w_{k,t}^m + \alpha^{-1}w_{k,t}^{\text{md},m} - w^*\|^2 + \gamma^2(1+\frac{1}{2}\alpha^{-1})\|\nabla F(w_{k,t}^{\text{md},m})\|^2 \\
&- 2\gamma(1+\frac{1}{2}\alpha^{-1})\langle(1-\alpha^{-1})w_{k,t}^m + \alpha^{-1}w_{k,t}^{\text{md},m} - w^*, \nabla F(w_{k,t}^{\text{md},m})\rangle + \gamma^2\sigma^2 \\
&\le (1+\frac{1}{2}\alpha^{-1})\Big((1-\alpha^{-1})\|w_{k,t}^m - w^*\|^2 + \alpha^{-1}\|w_{k,t}^{\text{md},m} - w^*\|^2\Big) + \gamma^2(1+\frac{1}{2}\alpha^{-1}) \\
&\cdot \|\nabla F(w_{k,t}^{\text{md},m})\|^2 - 2\gamma(1+\frac{1}{2}\alpha^{-1})\langle(1-\alpha^{-1})w_{k,t}^m + \alpha^{-1}w_{k,t}^{\text{md},m} - w^*, \nabla F(w_{k,t}^{\text{md},m})\rangle + \gamma^2\sigma^2
\end{aligned}
$$

It is easy to show $(1+\frac{1}{2}\alpha^{-1})(1-\alpha^{-1}) < 1 - \frac{1}{2}\alpha^{-1}, 1+\frac{1}{2}\alpha^{-1} \le \frac{3}{2}$. Due to these facts, we obtain

$$
\begin{aligned}
&\mathbb{E}[\|w_{k,t+1}^m - w^*\|^2|\mathcal{F}_{k,t}] \\
&\le (1-\frac{1}{2}\alpha^{-1})\|w_{k,t}^m - w^*\|^2 + \frac{3}{2}\alpha^{-1}\|w_{k,t}^{\text{md},m} - w^*\|^2 + \frac{3}{2}\gamma^2\|\nabla F(w_{k,t}^{\text{md},m})\|^2 \\
&- 2\gamma(1+\frac{1}{2}\alpha^{-1})\langle(1-\alpha^{-1})w_{k,t}^m + \alpha^{-1}w_{k,t}^{\text{md},m} - w^*, \nabla F(w_{k,t}^{\text{md},m})\rangle + \gamma^2\sigma^2 \\
&\le (1-\frac{1}{2}\alpha^{-1})\|w_{k,t}^m - w^*\|^2 + \frac{3}{2}\alpha^{-1}\|w_{k,t}^{\text{md},m} - w^*\|^2 + \frac{3}{2}\gamma^2\|\nabla F(w_{k,t}^{\text{md},m})\|^2 \\
&- 2\gamma(1+\frac{1}{2}\alpha^{-1})\langle(1-\alpha^{-1}(1-\beta^{-1}))w_{k,t}^m + \alpha^{-1}(1-\beta^{-1})w_{k,t}^{\text{ag},m} - w^*, \nabla F(w_{k,t}^{\text{md},m})\rangle + \gamma^2\sigma^2
\end{aligned}
$$

Next, we compute the upper bound of $\mathbb{E}[F(w_{k,t+1}^{\text{ag},m}) - F^*|\mathcal{F}_{k,t}]$.

$$
\begin{aligned}
&\mathbb{E}[F(w_{k,t+1}^{\text{ag},m}) - F^*|\mathcal{F}_{k,t}] \\
&\le \mathbb{E}[F(w_{k,t}^{\text{md},m}) + \langle\nabla F(w_{k,t}^{\text{md},m}), w_{k,t+1}^{\text{ag},m} - w_{k,t}^{\text{md},m}\rangle + \frac{L}{2}\|w_{k,t+1}^{\text{ag},m} - w_{k,t}^{\text{md},m}\|^2 - F^*|\mathcal{F}_{k,t}] \\
&\le F(w_{k,t}^{\text{md},m}) - F^* - \eta\|\nabla F(w_{k,t}^{\text{md},m})\|^2 + \frac{\eta^2 L}{2}\|\nabla F(w_{k,t}^{\text{md},m})\|^2 + \frac{\eta^2 L}{2}\sigma^2 \\
&\le F(w_{k,t}^{\text{md},m}) - F^* - \frac{\eta}{2}\|\nabla F(w_{k,t}^{\text{md},m})\|^2 + \frac{\eta^2 L}{2}\sigma^2 \ (\because 1 - \frac{\eta L}{2} \ge \frac{1}{2} \leftarrow \eta \in [0, \frac{1}{L}]) \\
&= (1-\frac{1}{2}\alpha^{-1})(F(w_{k,t}^{\text{ag},m}) - F^*) + \frac{1}{2}\alpha^{-1}(F(w_{k,t}^{\text{md},m}) - F^*) \\
&+ (1-\frac{1}{2}\alpha^{-1})(F(w_{k,t}^{\text{md},m}) - F(w_{k,t}^{\text{ag},m})) - \frac{\eta}{2}\|\nabla F(w_{k,t}^{\text{md},m})\|^2 + \frac{\eta^2 L}{2}\sigma^2 \\
&\le (1-\frac{1}{2}\alpha^{-1})(F(w_{k,t}^{\text{ag},m}) - F^*) - \frac{\mu\alpha^{-1}}{4}\|w_{k,t}^{\text{md},m} - w^*\|^2 + \frac{1}{2}\alpha^{-1}\langle\nabla F(w_{k,t}^{\text{md},m}), w_{k,t}^{\text{md},m} - w^*\rangle \\
&+ (1-\frac{1}{2}\alpha^{-1})\langle\nabla F(w_{k,t}^{\text{md},m}), w_{k,t}^{\text{md},m} - w_{k,t}^{\text{ag},m}\rangle - \frac{\eta}{2}\|\nabla F(w_{k,t}^{\text{md},m})\|^2 + \frac{\eta^2 L}{2}\sigma^2 \\
&= (1-\frac{1}{2}\alpha^{-1})(F(w_{k,t}^{\text{ag},m}) - F^*) - \frac{\mu\alpha^{-1}}{4}\|w_{k,t}^{\text{md},m} - w^*\|^2 - \frac{\eta}{2}\|\nabla F(w_{k,t}^{\text{md},m})\|^2 + \frac{\eta^2 L}{2}\sigma^2 \\
&+ \frac{1}{2}\alpha^{-1}\langle\nabla F(w_{k,t}^{\text{md},m}), 2\alpha\beta^{-1}w_{k,t}^m + (1-2\alpha\beta^{-1})w_{k,t}^{\text{ag},m} - w^*\rangle
\end{aligned}
$$

It is easy to show $\frac{1}{2}\alpha^{-1} = \frac{\gamma\mu}{3}(1+\frac{1}{2}\alpha^{-1})$. Then, we bound $\mathbb{E}[\Phi^m_{k,t+1}|\mathcal{F}_{k,t}]$ by using the above results.

$$\begin{aligned}
\mathbb{E}[\Phi^m_{k,t+1}|\mathcal{F}_{k,t}] &= \frac{\mu}{6}\mathbb{E}[\|w^m_{k,t+1} - w^*\|^2|\mathcal{F}_{k,t}] + \mathbb{E}[F(w^{\text{ag},m}_{k,t+1}) - F^*|\mathcal{F}_{k,t}] \\
&\leq (1-\frac{1}{2}\alpha^{-1})\Phi^m_{k,t} - \frac{2\eta-\gamma^2\mu}{4}\|\nabla F(w^{\text{md},m}_{k,t})\|^2 + \frac{1}{2}(\frac{\gamma^2\mu}{3}+\eta^2 L)\sigma^2 \\
&\leq (1-\frac{1}{2}\alpha^{-1})\Phi^m_{k,t} + \frac{1}{2}(\frac{\gamma^2\mu}{3}+\eta^2 L)\sigma^2 \ (\because \gamma \leq \sqrt{\frac{\eta}{\mu}}) \\
&\leq (1-\frac{1}{2}\alpha^{-1})\Phi^m_{k,t} + \frac{\gamma^2}{2}(\frac{\mu}{3}+L)\sigma^2 \qquad (55)
\end{aligned}$$

Plugging (54), (55) in (52) yields,

$$\begin{aligned}
&\mathbb{E}[B^m_{k,t+1}|\mathcal{F}_{k,t}] \\
&\leq \Big(\frac{\mu\alpha^{-2}}{3}(1-\beta^{-1})^2 + L\beta^{-2}\Big)\Big((1-\alpha^{-1})^2(1+2\alpha^{-1})\|w^m_{k,t} - w^{\text{md},m}_{k,t}\|^2 \\
&+ (\gamma-\eta)^2(1+\frac{\alpha}{2})\cdot(\frac{2\alpha^2-\alpha}{2\alpha^2-1}\cdot 2L\Phi^m_{k,t}) + (\gamma-\eta)^2\sigma^2\Big) \\
&+ \gamma^2(\frac{\mu}{3}+L)\frac{2\alpha^2-\alpha}{2\alpha^2-1}\cdot 2L\Big((1-\frac{1}{2}\alpha^{-1})\Phi^m_{k,t} + \frac{\gamma^2}{2}(\frac{\mu}{3}+L)\sigma^2\Big) \\
&= (1-\alpha^{-1})^2(1+2\alpha^{-1})\Big(\frac{\mu\alpha^{-2}}{3}(1-\beta^{-1})^2 + L\beta^{-2}\Big)\|w^m_{k,t} - w^{\text{md},m}_{k,t}\|^2 \\
&+ \Bigg(\Big(\frac{\mu\alpha^{-2}}{3}(1-\beta^{-1})^2 + L\beta^{-2}\Big)(\gamma-\eta)^2(1+\frac{\alpha}{2}) + (1-\frac{1}{2}\alpha^{-1})\gamma^2(\frac{\mu}{3}+L)\Bigg) \\
&\cdot(\frac{2\alpha^2-\alpha}{2\alpha^2-1}\cdot 2L\Phi^m_{k,t}) + \Bigg(\Big(\frac{\mu\alpha^{-2}}{3}(1-\beta^{-1})^2 + L\beta^{-2}\Big)(\gamma-\eta)^2 + \gamma^4(\frac{\mu}{3}+L)^2\frac{2\alpha^2-\alpha}{2\alpha^2-1}L\Bigg)\sigma^2 \qquad (56)
\end{aligned}$$

We can show that both coefficients of $\|w^m_{k,t} - w^{\text{md},m}_{k,t}\|^2$ and $\frac{2\alpha^2-\alpha}{2\alpha^2-1}\cdot 2L\Phi^m_{k,t}$ are upper bounded by $1-\frac{1}{2}\alpha^{-1} + \frac{\frac{1}{2}\alpha^{-1}}{1+\frac{1}{2}\alpha^{-1}}$.

$$\begin{aligned}
&(1-\alpha^{-1})^2(1+2\alpha^{-1}) \leq 1 - \frac{1}{2}\alpha^{-1} + \frac{\frac{1}{2}\alpha^{-1}}{1+\frac{1}{2}\alpha^{-1}} (< 1) \qquad (57) \\
\Leftrightarrow\ & 1 - \frac{1}{4}\alpha^{-2} + \frac{1}{2}\alpha^{-1} - (1-\alpha^{-1})^2(1+2\alpha^{-1})(1+\frac{1}{2}\alpha^{-1}) \geq 0
\end{aligned}$$

Let's define $g_1(\alpha^{-1}) = 1 - \frac{1}{4}\alpha^{-2} + \frac{1}{2}\alpha^{-1} - (1-\alpha^{-1})^2(1+2\alpha^{-1})(1+\frac{1}{2}\alpha^{-1})$. Then, it is easy to check that $g_1(\alpha^{-1}) \geq 0$ for $0 < \alpha^{-1} \leq 1$. Moreover, we would like to show the below inequality.

$$\begin{aligned}
&\Big(\frac{\mu\alpha^{-2}}{3}(1-\beta^{-1})^2 + L\beta^{-2}\Big)(\gamma-\eta)^2(1+\frac{\alpha}{2}) + (1-\frac{1}{2}\alpha^{-1})\gamma^2(\frac{\mu}{3}+L) \\
\leq\ & \Big(\frac{\mu\alpha^{-2}}{3}(1-\beta^{-1})^2 + L\beta^{-2}\Big)\gamma^2(1+\frac{\alpha}{2}) + (1-\frac{1}{2}\alpha^{-1})\gamma^2(\frac{\mu}{3}+L) \\
\leq\ & (1-\frac{1}{2}\alpha^{-1} + \frac{\frac{1}{2}\alpha^{-1}}{1+\frac{1}{2}\alpha^{-1}})\gamma^2(\frac{\mu}{3}+L) \qquad (58)
\end{aligned}$$

Since $\frac{\mu\alpha^{-2}}{3}(1-\beta^{-1})^2 + L\beta^{-2} = \frac{\mu}{3}(\frac{2\alpha-1}{2\alpha^2-1})^2 + L(\frac{\alpha-1}{2\alpha^2-1})^2 \leq (\frac{\mu}{3}+\frac{L}{4})(\frac{2\alpha-1}{2\alpha^2-1})^2$, it is enough to show

$$(\frac{\mu}{3}+\frac{L}{4})(\frac{2\alpha-1}{2\alpha^2-1})^2\gamma^2(1+\frac{\alpha}{2}) \leq \frac{\frac{1}{2}\alpha^{-1}}{1+\frac{1}{2}\alpha^{-1}}\gamma^2(\frac{\mu}{3}+L)$$

We also know that $\frac{\frac{\mu}{3}+L}{\frac{\mu}{3}+\frac{L}{4}} = 4 - \frac{1}{\frac{1}{3}+\frac{L}{\mu}\cdot\frac{1}{4}} > \frac{16}{7}$ ($\because \frac{L}{\mu} > 1$). Then, we only need to show

$$
\begin{aligned}
&(\frac{2\alpha-1}{2\alpha^2-1})^2(1+\frac{\alpha}{2}) \leq \frac{16}{7}\cdot\frac{\frac{1}{2}}{\alpha+\frac{1}{2}} \\
\Leftrightarrow\ &\frac{8}{7}(2\alpha^2-1)^2 - (2\alpha-1)^2(1+\frac{\alpha}{2})(\alpha+\frac{1}{2}) \geq 0
\end{aligned}
$$

Let's define $g_2(\alpha) = \frac{8}{7}(2\alpha^2-1)^2 - (2\alpha-1)^2(1+\frac{\alpha}{2})(\alpha+\frac{1}{2})$. Then, it is easy to check $g_2(\alpha) \geq 0$ for $\alpha \geq \frac{3}{2}$. As we assume $\gamma\mu \leq \frac{3}{4}$, we can say $\alpha = \frac{3}{2\gamma\mu} - \frac{1}{2} \geq \frac{3}{2}$. This indicates that the inequality (58) is satisfied. Thus, from (56), (57), and (58) we finally get

$$
\begin{aligned}
\mathbb{E}[B_{k,t+1}^m|\mathcal{F}_{k,t}] \leq (1 - \frac{1}{2}\alpha^{-1} + \frac{\frac{1}{2}\alpha^{-1}}{1+\frac{1}{2}\alpha^{-1}})B_{k,t}^m \\
+ \bigg(\Big(\frac{\mu\alpha^{-2}}{3}(1-\beta^{-1})^2 + L\beta^{-2}\Big)(\gamma-\eta)^2 + \gamma^4(\frac{\mu}{3}+L)^2\frac{2\alpha^2-\alpha}{2\alpha^2-1}L\bigg)\sigma^2
\end{aligned}
$$

From this relationship between $B_{k,t+1}^m$ and $B_{k,t}^m$, we obtain the result of Proposition 21.

$$
\begin{aligned}
\mathbb{E}[B_{k,t}^m] &\leq (1 - \frac{1}{2}\alpha^{-1} + \frac{\frac{1}{2}\alpha^{-1}}{1+\frac{1}{2}\alpha^{-1}})^t\mathbb{E}[B_{k,0}^m] + \bigg(\Big(\frac{\mu\alpha^{-2}}{3}(1-\beta^{-1})^2 + L\beta^{-2}\Big)(\gamma-\eta)^2 \\
&\quad + \gamma^4(\frac{\mu}{3}+L)^2\frac{2\alpha^2-\alpha}{2\alpha^2-1}L\bigg)\sigma^2 \cdot \frac{1-(1-\frac{1}{2}\alpha^{-1}+\frac{\frac{1}{2}\alpha^{-1}}{1+\frac{1}{2}\alpha^{-1}})^t}{1-(1-\frac{1}{2}\alpha^{-1}+\frac{\frac{1}{2}\alpha^{-1}}{1+\frac{1}{2}\alpha^{-1}})} \\
&\leq \mathbb{E}[B_{k,0}^m] + \bigg(\Big(\frac{\mu}{3}(\frac{2\alpha-1}{2\alpha^2-1})^2 + L(\frac{\alpha-1}{2\alpha^2-1})^2\Big)\cdot(\gamma-\eta)^2 + \gamma^4(\frac{\mu}{3}+L)^2\frac{2\alpha^2-\alpha}{2\alpha^2-1}L\bigg) \\
&\quad \cdot \frac{1+\frac{1}{2}\alpha^{-1}}{\frac{1}{4}\alpha^{-2}} \cdot \Big(1-(1-\frac{1}{2}\alpha^{-1}+\frac{\frac{1}{2}\alpha^{-1}}{1+\frac{1}{2}\alpha^{-1}})^t\Big)\sigma^2
\end{aligned}
$$

Proposition 22: Let F be μ-strongly convex, and assume Assumption 2, 3, 4, then for $\alpha = \frac{3}{2\gamma\mu} - \frac{1}{2}, \beta = \frac{2\alpha^2-1}{\alpha-1}, \gamma \in [\eta, \sqrt{\frac{\eta}{\mu}}], \eta, \gamma \in (0, \frac{1}{L}], \gamma\mu \leq \frac{3}{4}, \tau \geq 2$, FedAQ yields

$$
\begin{aligned}
\frac{\mu}{6}\mathbb{E}[\|w_{k,\tau}^m - w_k\|^2] + \frac{L}{2}\mathbb{E}[\|w_{k,\tau}^{\text{ag},m} - w_k^{\text{ag}}\|^2] \leq \Big(\gamma^2\mu(\frac{8}{3}\mu+2L) + 2\gamma^2L(\frac{\mu}{3}+L)\Big)\tau^2\mathbb{E}[\Phi_k] \\
+ (\frac{\gamma^2\mu}{3} + \eta^2L)\tau\sigma^2 \\
+ \Big((\gamma-\eta)^2\gamma^2\mu^2(\frac{\mu}{3}+\frac{L}{4}) + \gamma^4(\frac{\mu}{3}+L)^2L\Big)\frac{\tau^3\sigma^2}{2}
\end{aligned}
$$

Proof of Proposition 22 We use the same upper bounds for $\mathbb{E}[\|w_{k,\tau}^m - w_k\|^2]$ and $\mathbb{E}[\|w_{k,\tau}^{\text{ag},m} - w_k^{\text{ag}}\|^2]$ as in Proposition 15.

$$
\begin{aligned}
\mathbb{E}[\|w_{k,\tau}^m - w_k\|^2] &\leq \tau\Big(\sum_{t=0}^{\tau-1} 2\alpha^{-2}(1-\beta^{-1})^2\mathbb{E}[\|w_{k,t}^m - w_{k,t}^{\text{ag},m}\|^2] + 2\gamma^2\mathbb{E}[\|\nabla F(w_{k,t}^{\text{md},m})\|^2]\Big) \\
&\quad + 2\tau\gamma^2\sigma^2 \\
\mathbb{E}[\|w_{k,\tau}^{\text{ag},m} - w_k^{\text{ag}}\|^2] &\leq \tau\Big(\sum_{t=0}^{\tau-1} 2\beta^{-2}\mathbb{E}[\|w_{k,t}^m - w_{k,t}^{\text{ag},m}\|^2] + 2\eta^2\mathbb{E}[\|\nabla F(w_{k,t}^{\text{md},m})\|^2]\Big) + 2\tau\eta^2\sigma^2
\end{aligned}
$$

Thus, by using the above results, we get

$$
\begin{aligned}
&\frac{\mu}{6}\mathbb{E}[\|w_{k,\tau}^m - w_k\|^2] + \frac{L}{2}\mathbb{E}[\|w_{k,\tau}^{\mathrm{ag},m} - w_k^{\mathrm{ag}}\|^2] \\
&\leq \tau \sum_{t=0}^{\tau-1} \Big\{ \Big(\frac{\mu\alpha^{-2}}{3}(1-\beta^{-1})^2 + L\beta^{-2} \Big) \mathbb{E}[\|w_{k,t}^m - w_{k,t}^{\mathrm{ag},m}\|^2] + (\frac{\gamma^2\mu}{3} + \eta^2 L)\mathbb{E}[\|\nabla F(w_{k,t}^{\mathrm{md},m})\|^2] \Big\} \\
&+ (\frac{\gamma^2\mu}{3} + \eta^2 L)\tau\sigma^2 \\
&\leq \tau \sum_{t=0}^{\tau-1} \Big\{ \Big(\frac{\mu\alpha^{-2}}{3}(1-\beta^{-1})^2 + L\beta^{-2} \Big) \mathbb{E}[\|w_{k,t}^m - w_{k,t}^{\mathrm{ag},m}\|^2] + \gamma^2(\frac{\mu}{3} + L)\frac{2\alpha^2-\alpha}{2\alpha^2-1} 2L\mathbb{E}[\Phi_{k,t}^m] \Big\} \\
&+ (\frac{\gamma^2\mu}{3} + \eta^2 L)\tau\sigma^2 \ (\because (53)) \\
&= \tau\Big(\sum_{t=0}^{\tau-1} \mathbb{E}[B_{k,t}^m] \Big) + (\frac{\gamma^2\mu}{3} + \eta^2 L)\tau\sigma^2
\end{aligned}
$$

By Proposition 21 and the fact $\Phi_{k,0}^m = \Phi_k$, we obtain

$$
\begin{aligned}
&\frac{\mu}{6}\mathbb{E}[\|w_{k,\tau}^m - w_k\|^2] + \frac{L}{2}\mathbb{E}[\|w_{k,\tau}^{\mathrm{ag},m} - w_k^{\mathrm{ag}}\|^2] \\
&\leq \tau\Big\{ \sum_{t=0}^{\tau-1} \mathbb{E}[B_{k,0}^m] + \Big(\Big(\frac{\mu}{3}(\frac{2\alpha-1}{2\alpha^2-1})^2 + L(\frac{\alpha-1}{2\alpha^2-1})^2 \Big)(\gamma-\eta)^2 + \gamma^4(\frac{\mu}{3}+L)^2\frac{2\alpha^2-\alpha}{2\alpha^2-1}L \Big) \\
&\frac{1+\frac{1}{2}\alpha^{-1}}{\frac{1}{4}\alpha^{-2}} \Big(1 - (1 - \frac{1}{2}\alpha^{-1} + \frac{\frac{1}{2}\alpha^{-1}}{1+\frac{1}{2}\alpha^{-1}})^t \Big)\sigma^2 \Big\} + (\frac{\gamma^2\mu}{3} + \eta^2 L)\tau\sigma^2 \\
&= \tau^2\Big(\Big(\frac{\mu\alpha^{-2}}{3}(1-\beta^{-1})^2 + L\beta^{-2} \Big) \mathbb{E}[\|w_k - w_k^{\mathrm{ag}}\|^2] + \gamma^2(\frac{\mu}{3}+L)\frac{2\alpha^2-\alpha}{2\alpha^2-1} \cdot 2L\mathbb{E}[\Phi_k] \Big) \\
&+ \tau\Big(\Big(\frac{\mu}{3}(\frac{2\alpha-1}{2\alpha^2-1})^2 + L(\frac{\alpha-1}{2\alpha^2-1})^2 \Big) \cdot (\gamma-\eta)^2 + \gamma^4(\frac{\mu}{3}+L)^2\frac{2\alpha^2-\alpha}{2\alpha^2-1}L \Big) \frac{1+\frac{1}{2}\alpha^{-1}}{\frac{1}{4}\alpha^{-2}} \\
&\cdot \Big(\sum_{t=0}^{\tau-1} 1 - (1 - \frac{1}{2}\alpha^{-1} + \frac{\frac{1}{2}\alpha^{-1}}{1+\frac{1}{2}\alpha^{-1}})^t \Big)\sigma^2 + (\frac{\gamma^2\mu}{3} + \eta^2 L)\tau\sigma^2
\end{aligned}
$$

Before we get to the final result, let's find the upper bound for $\|w_k - w_k^{\mathrm{ag}}\|^2$, $\sum_{t=0}^{\tau-1}\Big(1 - (1 - \frac{1}{2}\alpha^{-1} + \frac{\frac{1}{2}\alpha^{-1}}{1+\frac{1}{2}\alpha^{-1}})^t\Big)$

$$
\begin{aligned}
\|w_k - w_k^{\mathrm{ag}}\|^2 &= \|w_k - w^* - (w_k^{\mathrm{ag}} - w^*)\|^2 \\
&\leq (1 + \frac{1}{3})\|w_k - w^*\|^2 + (1+3)\|w_k^{\mathrm{ag}} - w^*\|^2 \\
&\leq \frac{4}{3}\|w_k - w^*\|^2 + 4 \cdot \frac{2}{\mu}\Big(F(w_k^{\mathrm{ag}}) - F^* - \langle \nabla F(w^*), w_k^{\mathrm{ag}} - w^* \rangle \Big) \\
&= \frac{4}{3}\|w_k - w^*\|^2 + \frac{8}{\mu}(F(w_k^{\mathrm{ag}}) - F^*) = \frac{8}{\mu}\Phi_k
\end{aligned}
$$

$$\begin{aligned}
\sum_{t=0}^{\tau-1}\Big(1-(1-\frac{1}{2}\alpha^{-1}+\frac{\frac{1}{2}\alpha^{-1}}{1+\frac{1}{2}\alpha^{-1}})^t\Big) &= \tau-\sum_{t=0}^{\tau-1}(1-\frac{1}{2}\alpha^{-1}+\frac{\frac{1}{2}\alpha^{-1}}{1+\frac{1}{2}\alpha^{-1}})^t \\
&= \tau-\frac{1-(1-\frac{1}{2}\alpha^{-1}+\frac{\frac{1}{2}\alpha^{-1}}{1+\frac{1}{2}\alpha^{-1}})^\tau}{1-(1-\frac{1}{2}\alpha^{-1}+\frac{\frac{1}{2}\alpha^{-1}}{1+\frac{1}{2}\alpha^{-1}})} \\
&\le \tau-\frac{1-(1-\frac{\frac{1}{4}\alpha^{-2}}{1+\frac{1}{2}\alpha^{-1}}\tau+(\frac{\frac{1}{4}\alpha^{-2}}{1+\frac{1}{2}\alpha^{-1}})^2\frac{\tau(\tau-1)}{2})}{\frac{\frac{1}{4}\alpha^{-2}}{1+\frac{1}{2}\alpha^{-1}}} \\
&= \frac{\frac{1}{4}\alpha^{-2}}{1+\frac{1}{2}\alpha^{-1}}\cdot\frac{\tau(\tau-1)}{2}\le\frac{\frac{1}{4}\alpha^{-2}}{1+\frac{1}{2}\alpha^{-1}}\cdot\frac{\tau^2}{2}
\end{aligned}$$

Therefore, we obtain

$$\begin{aligned}
&\frac{\mu}{6}\mathbb{E}[\|w_{k,\tau}^m-w_k\|^2]+\frac{L}{2}\mathbb{E}[\|w_{k,\tau}^{\mathrm{ag},m}-w_k^{\mathrm{ag}}\|^2] \\
&\le\Big(\frac{8}{3}\alpha^{-2}(1-\beta^{-1})^2+\frac{8L}{\mu}\beta^{-2}+\gamma^2(\frac{\mu}{3}+L)\frac{2\alpha^2-\alpha}{2\alpha^2-1}\cdot 2L\Big)\tau^2\mathbb{E}[\Phi_k]+(\frac{\gamma^2\mu}{3}+\eta^2L)\tau\sigma^2 \\
&+\left(\Big(\frac{\mu}{3}(\frac{2\alpha-1}{2\alpha^2-1})^2+L(\frac{\alpha-1}{2\alpha^2-1})^2\Big)\cdot(\gamma-\eta)^2+\gamma^4(\frac{\mu}{3}+L)^2\frac{2\alpha^2-\alpha}{2\alpha^2-1}L\right)\cdot\frac{\tau^3\sigma^2}{2} \qquad (59)
\end{aligned}$$

Moreover, we can simplify the above inequality by replacing α,β with γ,μ. It is easy to show $\frac{2\alpha^2-\alpha}{2\alpha^2-1}\le 1$, $\frac{2\alpha-1}{2\alpha^2-1}\le\frac{1}{\alpha}=\frac{2\gamma\mu}{3-\gamma\mu}\le\gamma\mu$. Then, we can further show

$$\begin{aligned}
&\frac{8}{3}\alpha^{-2}(1-\beta^{-1})^2+\frac{8L}{\mu}\beta^{-2}+\gamma^2(\frac{\mu}{3}+L)\frac{2\alpha^2-\alpha}{2\alpha^2-1}\cdot 2L \\
=&\frac{8}{3}(\frac{2\alpha-1}{2\alpha^2-1})^2+\frac{8L}{\mu}(\frac{\alpha-1}{2\alpha^2-1})^2+\gamma^2(\frac{\mu}{3}+L)\frac{2\alpha^2-\alpha}{2\alpha^2-1}\cdot 2L \\
\le&(\frac{8}{3}+\frac{2L}{\mu})(\frac{2\alpha-1}{2\alpha^2-1})^2+\gamma^2(\frac{\mu}{3}+L)2L \\
\le&(\frac{8}{3}+\frac{2L}{\mu})\alpha^{-2}+\gamma^2(\frac{\mu}{3}+L)2L \\
\le&\gamma^2\mu(\frac{8}{3}\mu+2L)+2\gamma^2L(\frac{\mu}{3}+L) \qquad (60)
\end{aligned}$$

We also get

$$\begin{aligned}
&\Big(\frac{\mu}{3}(\frac{2\alpha-1}{2\alpha^2-1})^2+L(\frac{\alpha-1}{2\alpha^2-1})^2\Big)\cdot(\gamma-\eta)^2+\gamma^4(\frac{\mu}{3}+L)^2\frac{2\alpha^2-\alpha}{2\alpha^2-1}L \\
\le&(\frac{\mu}{3}+\frac{L}{4})(\frac{2\alpha-1}{2\alpha^2-1})^2(\gamma-\eta)^2+\gamma^4(\frac{\mu}{3}+L)^2L \\
\le&(\gamma-\eta)^2\gamma^2\mu^2(\frac{\mu}{3}+\frac{L}{4})+\gamma^4(\frac{\mu}{3}+L)^2L \qquad (61)
\end{aligned}$$

Finally, from (59), (60), and (61), we conclude as below

$$\begin{aligned}\frac{\mu}{6}\mathbb{E}[\|w_{k,\tau}^m - w_k\|^2] + \frac{L}{2}\mathbb{E}[\|w_{k,\tau}^{\mathrm{ag},m} - w_k^{\mathrm{ag}}\|^2] \leq \Big(\gamma^2\mu(\frac{8}{3}\mu + 2L) + 2\gamma^2 L(\frac{\mu}{3} + L)\Big)\tau^2\mathbb{E}[\Phi_k] \\ + (\frac{\gamma^2\mu}{3} + \eta^2 L)\tau\sigma^2 \\ + \Big((\gamma-\eta)^2\gamma^2\mu^2(\frac{\mu}{3} + \frac{L}{4}) + \gamma^4(\frac{\mu}{3} + L)^2 L\Big)\frac{\tau^3\sigma^2}{2}\end{aligned}$$

Proof of Lemma 5 By the definition of $\Phi_k, \Phi_{k,t}$ and Proposition 18,

$$\mathbb{E}[\Phi_{k+1}] = \mathbb{E}[\Phi_{k,\tau}] + \frac{\mu}{6}\mathbb{E}[\|w_{k+1} - \bar{w}_{k,\tau}\|^2] + \mathbb{E}[F(w_{k+1}^{\mathrm{ag}}) - F(\bar{w}_{k,\tau}^{\mathrm{ag}})]$$

Applying Proposition 19 and Proposition 20, we have

$$\begin{aligned}&\mathbb{E}[\Phi_{k+1}] \\ &\leq (1 - \frac{1}{3}\gamma\mu)^\tau\mathbb{E}[\Phi_k] + (\frac{\eta^2 L}{2} + \frac{\gamma^2\mu}{6})\frac{\tau\sigma^2}{M} + \gamma\tau\cdot\max_{0\leq t<\tau}\mathbb{E}[\|\nabla F(\bar{w}_{k,t}^{\mathrm{md}}) - \frac{1}{M}\sum_{m=1}^M \nabla F(w_{k,t}^{\mathrm{md},m})\|^2] \\ &+ \frac{q\mu}{6M^2}\sum_{m=1}^M \mathbb{E}[\|w_{k,\tau}^m - w_k\|^2] + \frac{qL}{2M^2}\sum_{m=1}^M \mathbb{E}[\|w_{k,\tau}^{\mathrm{ag},m} - w_k^{\mathrm{ag}}\|^2] \\ &\leq (1 - \frac{1}{3}\gamma\mu)^\tau\mathbb{E}[\Phi_k] + (\frac{\eta^2 L}{2} + \frac{\gamma^2\mu}{6})\frac{\tau\sigma^2}{M} + \gamma\tau\cdot\max_{0\leq t<\tau}\mathbb{E}[\|\nabla F(\bar{w}_{k,t}^{\mathrm{md}}) - \frac{1}{M}\sum_{m=1}^M \nabla F(w_{k,t}^{\mathrm{md},m})\|^2] \\ &+ \frac{q}{M}\Big[\Big(\gamma^2\mu(\frac{8}{3}\mu + 2L) + 2\gamma^2 L(\frac{\mu}{3} + L)\Big)\tau^2\mathbb{E}[\Phi_k] + (\frac{\gamma^2\mu}{3} + \eta^2 L)\tau\sigma^2 \\ &+ \Big((\gamma-\eta)^2\gamma^2\mu^2(\frac{\mu}{3} + \frac{L}{4}) + \gamma^4(\frac{\mu}{3} + L)^2 L\Big)\frac{\tau^3\sigma^2}{2}\Big] \\ &= D(\gamma,\tau)\mathbb{E}[\Phi_k] + (\frac{\eta^2 L}{2} + \frac{\gamma^2\mu}{6})\frac{\tau\sigma^2}{M} + \gamma\tau\cdot\max_{0\leq t<\tau}\mathbb{E}[\|\nabla F(\bar{w}_{k,t}^{\mathrm{md}}) - \frac{1}{M}\sum_{m=1}^M \nabla F(w_{k,t}^{\mathrm{md},m})\|^2] \\ &+ \frac{q}{M}(\frac{\gamma^2\mu}{3} + \eta^2 L)\tau\sigma^2 + \frac{q}{2M}\Big((\gamma-\eta)^2\gamma^2\mu^2(\frac{\mu}{3} + \frac{L}{4}) + \gamma^4(\frac{\mu}{3} + L)^2 L\Big)\tau^3\sigma^2\end{aligned}$$

The second inequality comes from Proposition 22. $D(\gamma,\tau)$ is defined as below.

$$D(\gamma,\tau) = (1 - \frac{1}{3}\gamma\mu)^\tau + \frac{q}{M}\Big(\gamma^2\mu(\frac{8}{3}\mu + 2L) + 2\gamma^2 L(\frac{\mu}{3} + L)\Big)\tau^2$$

5.4.2 Proof of Theorem 6

Proof of Theorem 6 At first, due to the condition (35) in Theorem 6, we get

$$\begin{aligned}D(\gamma,\tau) &= (1 - \frac{1}{3}\gamma\mu)^\tau + \frac{q}{M}\Big(\gamma^2\mu(\frac{8}{3}\mu + 2L) + 2\gamma^2 L(\frac{\mu}{3} + L)\Big)\tau^2 \\ &\leq 1 - \frac{1}{3}\gamma\mu\tau + \frac{1}{9}\gamma^2\mu^2\tau^2 + \frac{q}{M}\gamma^2\Big(\mu(\frac{8}{3}\mu + 2L) + 2L(\frac{\mu}{3} + L)\Big)\tau^2 \\ &= 1 - \frac{1}{3}\gamma\mu\tau + \Big(\frac{1}{9}\mu^2 + \frac{q}{M}\Big(\mu(\frac{8}{3}\mu + 2L) + 2L(\frac{\mu}{3} + L)\Big)\Big)\gamma^2\tau^2 \\ &\leq 1 - \frac{1}{6}\gamma\mu\tau \ (\because \text{ condition (35)})\end{aligned}$$

It is trivial that $\gamma = \max(\eta, \sqrt{\frac{\eta}{\mu\tau}}) \in [\eta, \sqrt{\frac{\eta}{\mu}}]$. Thus, we can use Lemma 5. By using Lemma 5 and the above result, we obtain

$$
\begin{aligned}
&\mathbb{E}[\Phi_{k+1}] \\
&\leq (1 - \frac{1}{6}\gamma\mu\tau)\mathbb{E}[\Phi_k] + (\frac{\eta^2 L}{2} + \frac{\gamma^2\mu}{6})\frac{\tau\sigma^2}{M} + \gamma\tau \cdot \max_{0\leq t<\tau} \mathbb{E}[\|\nabla F(\bar{w}_{k,t}^{\text{md}}) - \frac{1}{M}\sum_{m=1}^{M} \nabla F(w_{k,t}^{\text{md},m})\|^2] \\
&+ \frac{q}{M}(\frac{\gamma^2\mu}{3} + \eta^2 L)\tau\sigma^2 + \frac{q}{2M}\Big((\gamma-\eta)^2\gamma^2\mu^2(\frac{\mu}{3} + \frac{L}{4}) + \gamma^4(\frac{\mu}{3} + L)^2 L\Big)\tau^3\sigma^2 \qquad (62)
\end{aligned}
$$

By the Lemma C.14 in [43], we know that the below quantity is bounded.

$$
\begin{aligned}
&\max_{0\leq t<\tau} \mathbb{E}[\|\nabla F(\bar{w}_{k,t}^{\text{md}}) - \frac{1}{M}\sum_{m=1}^{M} \nabla F(w_{k,t}^{\text{md},m})\|^2] \leq B' \\
&B' = \begin{cases} 4\eta^2 L^2\tau\sigma^2\Big(1 + \frac{\gamma^2\mu}{\eta}\Big)^{2\tau}, & \text{if } \gamma \in \Big(\eta, \sqrt{\frac{\eta}{\mu}}\Big] \\ 4\eta^2 L^2\tau\sigma^2, & \text{if } \gamma = \eta \end{cases}
\end{aligned}
$$

Telescoping (62) yields

$$
\begin{aligned}
\mathbb{E}[\Phi_K] &\leq (1 - \frac{1}{6}\gamma\mu\tau)^K\Phi_0 + \Big(\sum_{k'=0}^{K-1}(1 - \frac{1}{6}\gamma\mu\tau)^{k'}\Big) \cdot \Big[(\frac{\eta^2 L}{2} + \frac{\gamma^2\mu}{6})\frac{\tau\sigma^2}{M} + \frac{q}{M}(\frac{\gamma^2\mu}{3} + \eta^2 L)\tau\sigma^2 \\
&\quad + \frac{q}{2M}\Big((\gamma-\eta)^2\gamma^2\mu^2(\frac{\mu}{3} + \frac{L}{4}) + \gamma^4(\frac{\mu}{3} + L)^2 L\Big)\tau^3\sigma^2 + \gamma\tau B'\Big] \\
&\leq \exp\Big(-\frac{\gamma\mu\tau K}{6}\Big)\Phi_0 + \frac{3\eta^2 L\sigma^2}{\gamma\mu M} + \frac{\gamma\sigma^2}{M} + \frac{6B'}{\mu} + 2q\Big(\frac{\gamma\sigma^2}{M} + \frac{3\eta^2 L\sigma^2}{\gamma\mu M}\Big) \\
&\quad + \frac{3q}{M}\Big((\gamma-\eta)^2\gamma\mu(\frac{\mu}{3} + \frac{L}{4}) + \frac{\gamma^3(\frac{\mu}{3} + L)^2 L}{\mu}\Big)\tau^2\sigma^2
\end{aligned}
$$

The last inequality comes from the fact that $\sum_{k'=0}^{K-1}(1 - \frac{1}{6}\gamma\mu\tau)^{k'} \leq \frac{6}{\gamma\mu\tau}$. Since we plug in $\gamma = \max(\eta, \sqrt{\frac{\eta}{\mu\tau}})$, we can use Lemma C.15 in [43]. Therefore, we obtain

$$
\begin{aligned}
\mathbb{E}[\Phi_K] &\leq \exp\Big(-\frac{1}{6}\max(\eta\mu, \sqrt{\frac{\eta\mu}{\tau}})K\tau\Big)\Phi_0 + \frac{2(2q+1)\eta^{\frac{1}{2}}\sigma^2}{\mu^{\frac{1}{2}}M\tau^{\frac{1}{2}}} + \frac{4(2q+1)\eta^2 L^2\tau\sigma^2}{\mu} \\
&\quad + \frac{24e^2\eta^2 L^2\tau\sigma^2}{\mu} + \frac{3q\tau^2\sigma^2}{M}\max\Big(\frac{\eta^{\frac{3}{2}}\mu(\frac{\mu}{3} + \frac{L}{4})}{\mu^{\frac{3}{2}}\tau^{\frac{3}{2}}} + \frac{\eta^{\frac{3}{2}}(\frac{\mu}{3} + L)^2 L}{\mu^{\frac{5}{2}}\tau^{\frac{3}{2}}}, \frac{\eta^3(\frac{\mu}{3} + L)^2 L}{\mu}\Big)
\end{aligned}
$$

The first term stems directly from Lemma C.15 in [43]. Also, the last term comes from the fact that

$$
(\gamma-\eta)^2\gamma\mu(\frac{\mu}{3} + \frac{L}{4}) + \frac{\gamma^3(\frac{\mu}{3} + L)^2 L}{\mu} \leq \begin{cases} \gamma^3\mu(\frac{\mu}{3} + \frac{L}{4}) + \frac{\gamma^3(\frac{\mu}{3}+L)^2 L}{\mu}, & \text{if } \gamma \neq \eta \\ \frac{\eta^3(\frac{\mu}{3}+L)^2 L}{\mu}, & \text{if } \gamma = \eta \end{cases}
$$

Therefore, by simple inequalities such as $\max(a, b) \leq a + b$ and $\min(a, b) \leq a$, we ultimately get

$$
\begin{aligned}
\mathbb{E}[\Phi_K] &\leq \exp\Big(-\frac{1}{6}\max(\eta\mu, \sqrt{\frac{\eta\mu}{\tau}})K\tau\Big)\Phi_0 + \frac{2(2q+1)\eta^{\frac{1}{2}}\sigma^2}{\mu^{\frac{1}{2}}M\tau^{\frac{1}{2}}} + \frac{8(q+25)\eta^2 L^2\tau\sigma^2}{\mu} \\
&\quad + \frac{3q\Big(\mu^2(\frac{\mu}{3} + \frac{L}{4}) + L(\frac{\mu}{3} + L)^2\Big)\eta^{\frac{3}{2}}\tau^{\frac{1}{2}}\sigma^2}{\mu^{\frac{5}{2}}M} + \frac{3qL(\frac{\mu}{3} + L)^2\eta^3\tau^2\sigma^2}{\mu M} \qquad (63)
\end{aligned}
$$

5.4.3 Proof of Corollary 23

Corollary 23: Let $D_1, D_2,$ and η_0 as below. Note that $T = K\tau$.

$$D_1 = \frac{\Big(\mu^2(\frac{\mu}{3}+\frac{L}{4}) + L(\frac{\mu}{3}+L)^2)\Big)q}{\mu^{\frac{5}{2}}},\ D_2 = \frac{q(\frac{\mu}{3}+L)^2L}{\mu}$$

$$\eta_0 = \frac{36\tau}{\mu T^2}\log^2\Big(e + \min(\frac{\mu MT\Phi_0}{(2q+1)\sigma^2}, \frac{\mu^3T^4\Phi_0}{(q+25)L^2\tau^3\sigma^2}, \frac{\mu^3MT^3\Phi_0}{(\mu^{\frac{3}{2}}D_1 + 6^3D_2)\tau^2\sigma^2})\Big)$$

Then for $\eta = \min(\frac{1}{L}, \eta_0)$, FedAQ yields

$$\begin{aligned}
\mathbb{E}[\Phi_K] \leq \min\Big(\exp(-\frac{\mu T}{6L}), \exp(-\frac{\mu^{\frac{1}{2}}T}{6L^{\frac{1}{2}}\tau^{\frac{1}{2}}})\Big)\Phi_0 \\
+ \frac{13(2q+1)\sigma^2}{\mu MT}\log^2\Big(e + \frac{\mu MT\Phi_0}{(2q+1)\sigma^2}\Big) \qquad (64)\\
+ \frac{10369(q+25)L^2\tau^3\sigma^2}{\mu^3T^4}\log^4\Big(e + \frac{\mu^3T^4\Phi_0}{(q+25)L^2\tau^3\sigma^2}\Big) \qquad (65)\\
+ \frac{649(\mu^{\frac{3}{2}}D_1 + 216D_2)\tau^2\sigma^2}{\mu^3MT^3}\log^6\Big(e + \frac{\mu^3MT^3\Phi_0}{(\mu^{\frac{3}{2}}D_1 + 216D_2)\tau^2\sigma^2}\Big) \qquad (66)
\end{aligned}$$

Proof of Corollary 23 Let's decompose the final result (63) of the Theorem 6 into a decreasing term and an increasing term. We denote the decreasing term ϕ_1 and the increasing term ϕ_2 as below.

$$\begin{aligned}
\phi_1(\eta) &= \exp\Big(-\frac{1}{6}\max(\eta\mu, \sqrt{\frac{\eta\mu}{\tau}})T\Big)\Phi_0\\
\phi_2(\eta) &= \frac{2(2q+1)\eta^{\frac{1}{2}}\sigma^2}{\mu^{\frac{1}{2}}M\tau^{\frac{1}{2}}} + \frac{8(q+25)\eta^2L^2\tau\sigma^2}{\mu} + \frac{3q\Big(\mu^2(\frac{\mu}{3}+\frac{L}{4}) + L(\frac{\mu}{3}+L)^2\Big)\eta^{\frac{3}{2}}\tau^{\frac{1}{2}}\sigma^2}{\mu^{\frac{5}{2}}M}\\
&\quad + \frac{3qL(\frac{\mu}{3}+L)^2\eta^3\tau^2\sigma^2}{\mu M}
\end{aligned}$$

Since ϕ_1 is the decreasing term, we have

$$\phi_1(\eta) \leq \phi_1(\frac{1}{L}) + \phi_1(\eta_0) \qquad (67)$$

where

$$\begin{aligned}
\phi_1(\frac{1}{L}) &= \min\Big(\exp(-\frac{\mu T}{6L}), \exp(-\frac{\mu^{\frac{1}{2}}T}{6L^{\frac{1}{2}}\tau^{\frac{1}{2}}})\Big)\Phi_0\\
\phi_1(\eta_0) &\leq \exp\Big(-\frac{1}{6}\sqrt{\frac{\eta_0\mu}{\tau}}T\Big)\\
&= \Big(e + \min(\frac{\mu MT\Phi_0}{(2q+1)\sigma^2}, \frac{\mu^3T^4\Phi_0}{(q+25)L^2\tau^3\sigma^2}, \frac{\mu^3MT^3\Phi_0}{(\mu^{\frac{3}{2}}D_1 + 6^3D_2)\tau^2\sigma^2})\Big)^{-1}\Phi_0\\
&\leq \frac{(2q+1)\sigma^2}{\mu MT} + \frac{(q+25)L^2\tau^3\sigma^2}{\mu^3T^4} + \frac{(\mu^{\frac{3}{2}}D_1 + 6^3D_2)\tau^2\sigma^2}{\mu^3MT^3}
\end{aligned}$$

Since ϕ_2 is the increasing term, we have

$$
\begin{aligned}
&\phi_2(\eta) \\
&\leq \phi_2(\eta_0) \\
&\leq \frac{12(2q+1)\sigma^2}{\mu MT}\log\Big(e+\frac{\mu MT\Phi_0}{(2q+1)\sigma^2}\Big)+\frac{8\cdot 36^2(q+25)L^2\tau^3\sigma^2}{\mu^3T^4}\log^4\Big(e+\frac{\mu^3T^4\Phi_0}{(q+25)L^2\tau^3\sigma^2}\Big) \\
&+\frac{3\cdot 6^3D_1\tau^2\sigma^2}{\mu^{\frac{3}{2}}MT^3}\log^3\Big(e+\frac{\mu^3MT^3\Phi_0}{(\mu^{\frac{3}{2}}D_1+6^3D_2)\tau^2\sigma^2}\Big) \\
&+\frac{3\cdot 36^3D_2\tau^5\sigma^2}{\mu^3MT^6}\log^6\Big(e+\frac{\mu^3MT^3\Phi_0}{(\mu^{\frac{3}{2}}D_1+6^3D_2)\tau^2\sigma^2}\Big) \\
&\leq \frac{12(2q+1)\sigma^2}{\mu MT}\log\Big(e+\frac{\mu MT\Phi_0}{(2q+1)\sigma^2}\Big)+\frac{8\cdot 36^2(q+25)L^2\tau^3\sigma^2}{\mu^3T^4}\log^4\Big(e+\frac{\mu^3T^4\Phi_0}{(q+25)L^2\tau^3\sigma^2}\Big) \\
&+\frac{3\cdot 6^3(\mu^{\frac{3}{2}}D_1+6^3D_2)\tau^2\sigma^2}{\mu^3MT^3}\log^6\Big(e+\frac{\mu^3MT^3\Phi_0}{(\mu^{\frac{3}{2}}D_1+6^3D_2)\tau^2\sigma^2}\Big) \qquad (68)
\end{aligned}
$$

The last inequality comes from $\frac{\tau}{T}\leq 1$. Therefore, by combining (67) and (68), we finally get

$$
\begin{aligned}
\mathbb{E}[\Phi_K] &\leq \phi_1(\eta)+\phi_2(\eta) \\
&\leq \phi_1(\frac{1}{L})+\phi_1(\eta_0)+\phi_2(\eta_0) \\
&\leq \min\Big(\exp(-\frac{\mu T}{6L}),\exp(-\frac{\mu^{\frac{1}{2}}T}{6L^{\frac{1}{2}}\tau^{\frac{1}{2}}})\Big)\Phi_0+\frac{13(2q+1)\sigma^2}{\mu MT}\log^2\Big(e+\frac{\mu MT\Phi_0}{(2q+1)\sigma^2}\Big) \\
&+\frac{10369(q+25)L^2\tau^3\sigma^2}{\mu^3T^4}\log^4\Big(e+\frac{\mu^3T^4\Phi_0}{(q+25)L^2\tau^3\sigma^2}\Big) \\
&+\frac{649(\mu^{\frac{3}{2}}D_1+216D_2)\tau^2\sigma^2}{\mu^3MT^3}\log^6\Big(e+\frac{\mu^3MT^3\Phi_0}{(\mu^{\frac{3}{2}}D_1+216D_2)\tau^2\sigma^2}\Big)
\end{aligned}
$$

5.5 More Theoretical Details about Remark 9 and Contribution 2 in Introduction

5.5.1 Why Haddadpour et al. (2021) Cannot Achieve a Linear Speedup

It is hard to say that [9] achieves a linear speedup in M in strongly-convex and homogeneous settings. Let's first recap Corollary D.8 in [9]. They let $\eta\gamma\mu\tau\leq\frac{1}{2},\kappa=\frac{L}{\mu},\gamma\geq M$ and tune η as $\eta=\frac{1}{2L(\frac{q}{M}+1)\tau\gamma}$. Here, η is the client learning rate, and γ is the server learning rate. Other parameters are the same as we defined. Then, they obtain the below result.

$$
\begin{aligned}
\mathbb{E}[F(w_K)-F^*] &\leq \exp(-\eta\gamma\mu\tau K)(F(w_0)-F^*)+\frac{1}{\mu}\Big[\frac{1}{2}\tau L^2\eta^2\sigma^2+(1+q)\frac{\gamma\eta L\sigma^2}{2M}\Big] \qquad (69) \\
&\leq \mathcal{O}\Big(\exp(-\frac{K}{2(\frac{q}{M}+1)\kappa})(F(w_0)-F^*)+\frac{\sigma^2}{\gamma^2\mu\tau}+\frac{(q+1)\sigma^2}{\mu(\frac{q}{M}+1)\tau M}\Big) \\
&= \mathcal{O}\Big(\exp(-\frac{K}{2(\frac{q}{M}+1)\kappa})(F(w_0)-F^*)+\frac{\sigma^2K}{\gamma^2\mu T}+\frac{(q+1)K\sigma^2}{\mu(\frac{q}{M}+1)TM}\Big)
\end{aligned}
$$

Let's focus on the second and third term. We assume M is large enough and represent them only with γ,K,T,M to easily check the linear speedup of this convergence rate. Then, we obtain

$$
\mathcal{O}\Big(\frac{K}{\gamma^2T}+\frac{K}{MT}\Big)\leq\mathcal{O}\Big(\frac{K}{M^2T}+\frac{K}{MT}\Big)\ (\because\gamma\geq M) \qquad (70)
$$

Thus, it seemingly achieves a linear speedup in M when K is just a constant. However, we are missing the critical point in this analysis. To be specific, let's consider the case when $\gamma = 1$. Then, the convergence rate (70) changes into $\mathcal{O}\Big(\frac{K}{T} + \frac{K}{MT}\Big)$ that cannot achieve a linear speedup in M. This is implausible because the convergence rate (69) becomes tighter when $\gamma = 1$ than $\gamma \geq M$ (See the last term of (69)). Actually, we can achieve a linear speedup in M when $\gamma = 1$ if we tune $\eta = \frac{1}{2L(\frac{q}{M}+1)\tau M}$. However, this is not an appropriate tuning because there is M in the denominator. Similarly, [9] tunes $\eta = \frac{1}{2L(\frac{q}{M}+1)\tau\gamma}$ where $\gamma \geq M$. Even though there is no M in the denominator, the condition $\gamma \geq M$ forcibly makes the convergence rate achieve a linear speedup without any theoretical benefits of the algorithm. Therefore, we cannot say their η makes their algorithm achieve a linear speedup in M. We should tune in a different way that does not contain M in a denominator. For reference, our tuning parameter η for the FedAQ algorithm does not contain M in the denominator (See Corollary 17 and Corollary 23).

5.5.2 New Convergence Rate for Haddadpour et al. (2021)

We propose new η and convergence rate for [9]. This new η makes the algorithm achieve a linear speedup in M. Let's denote $\Phi_0 = F(w_0) - F^*$. We also know that $T = K\tau$. Then, we choose η as

$$\eta = \frac{1}{\gamma\mu T}\log\Big(e + \min(\frac{\gamma^2\mu^3T^2\Phi_0}{\tau L^2\sigma^2}, \frac{\mu^2MT\Phi_0}{(1+q)L\sigma^2})\Big)$$

We plug in this η to (69). We bound the first term as below.

$$\begin{aligned}\exp(-\eta\gamma\mu\tau K)(F(w_0) - F^*) &= \Big(e + \min(\frac{\gamma^2\mu^3T^2\Phi_0}{\tau L^2\sigma^2}, \frac{\mu^2MT\Phi_0}{(1+q)L\sigma^2})\Big)^{-1}\Phi_0 \\ &\leq \frac{\tau L^2\sigma^2}{\gamma^2\mu^3T^2} + \frac{(1+q)L\sigma^2}{\mu^2MT}\end{aligned}$$

The another terms are bounded as below.

$$\begin{aligned}\frac{1}{\mu}\Big[\frac{1}{2}\tau L^2\eta^2\sigma^2 + (1+q)\frac{\gamma\eta L\sigma^2}{2M}\Big] &\leq \frac{\tau L^2\sigma^2}{2\gamma^2\mu^3T^2}\log^2\Big(e + \frac{\gamma^2\mu^3T^2\Phi_0}{\tau L^2\sigma^2}\Big) \\ &\quad + \frac{(1+q)L\sigma^2}{2\mu^2MT}\log\Big(e + \frac{\mu^2MT\Phi_0}{(1+q)L\sigma^2}\Big)\end{aligned}$$

Thus, we obtain a new convergence rate by combining the above two bounds.

$$\begin{aligned}\mathbb{E}[F(w_K) - F^*] &\leq \exp(-\eta\gamma\mu\tau K)(F(w_0) - F^*) + \frac{1}{\mu}\Big[\frac{1}{2}\tau L^2\eta^2\sigma^2 + (1+q)\frac{\gamma\eta L\sigma^2}{2M}\Big] \\ &\leq \frac{3\tau L^2\sigma^2}{2\gamma^2\mu^3T^2}\log^2\Big(e + \frac{\gamma^2\mu^3T^2\Phi_0}{\tau L^2\sigma^2}\Big) + \frac{3(1+q)L\sigma^2}{2\mu^2MT}\log\Big(e + \frac{\mu^2MT\Phi_0}{(1+q)L\sigma^2}\Big)\end{aligned}$$

Here, we replace τ with $\frac{T}{K}$. Then, we represent the above convergence rate with only T, K, M, q.

$$\tilde{\mathcal{O}}(\frac{1}{TK} + \frac{1+q}{MT})$$

This is the new convergence rate we propose in Remark 9. We also get $K = \tilde{\mathcal{O}}(\frac{M}{1+q})$ communication rounds make this algorithm achieve a linear speedup in M.

5.5.3 More Details on Contribution 2 in Introduction

More Details on d_{quant} This paragraph explains why FedAQ needs to send only $d_{\text{quant}} = O(\log \frac{1}{q})$ bits for each value. We use the result of Lemma 3.1 in [1]. They show the below result with a low-precision quantizer (Example 1 in section 3)

$$\mathbb{E}[\|Q(x,s) - x\|_2^2] \leq \min(\frac{n}{s^2}, \frac{\sqrt{n}}{s})\|x\|_2^2$$

where n is the dimension of x, and s is the number of quantization levels. Then, we regard q as

$$q = \frac{\sqrt{n}}{s} = \frac{\sqrt{n}}{2^{d_{\text{quant}}}} \tag{71}$$

Thus, we obtain the following conclusion.

$$d_{\text{quant}} = \frac{\frac{1}{2}\log n + \log \frac{1}{q}}{\log 2} = O(\log \frac{1}{q})$$

Comparing FedAQ to FedAC We compare computation and communication efficiency of FedAC-II and FedAQ under the condition set (34) to achieve the same error. Let's recall the convergence rate of FedAC and FedAQ. The convergence rate of FedAC and FedAQ is respectively $\tilde{\mathcal{O}}(\frac{1}{MT} + \frac{1}{TK^3})$ and $\tilde{\mathcal{O}}(\frac{1+q}{MT} + \frac{1+q}{TK^3})$. Let's say FedAC requires T iterations and $K = M^{\frac{1}{3}}$ communication rounds to achieve the error $\frac{1}{MT}$. Then, FedAQ requires

$$T' = (1+q)T,\ K' = M^{\frac{1}{3}}$$

to achieve the same error $\frac{1}{MT}$. This means FedAQ needs $1+q$ times more local steps and the same number of communication rounds to achieve the same error of FedAC. These local steps do no require any communication with the server hence can be performed without any additional communication overhead.

From discussion in the previous section, if we use the simple low-precision quantizer, we need only $d_{\text{quant}} = O(\log \frac{1}{q})$ bits for communicating values with enough precision that can lead to an error rate of $O(\frac{1}{MT})$. In comparison, FedAC would require $O(\log(MT))$ bits to maintain enough precision to achieve the same error rate. In a majority of tasks in the real world, 32 bits are usually enough for d_{full} to achieve enough precision as we usually don't need converge to a very small error rate. Nonetheless, even if we compare FedAQ(8bits) with to FedAC(32bits), we argue that the overall benefit from less communication by quantization is more influential than the slowdown effect from quantization.

For example, if we consider a l_2-regularized logistic regression model for MNIST (strongly convex experiment) and quantize from 32 bits to $d_{\text{quant}} = 8$ bits. Here, $n = 784 \times 10$. We get the following results by using (71).

$$1 + q = 1 + \frac{\sqrt{n}}{2^{d_{\text{quant}}}} = 1 + \frac{\sqrt{7840}}{2^8} \simeq 1.346,$$

On the other hand, the ratio of data communicated by FedAC and FedAQ is

$$\frac{32}{d_{\text{quant}}} = 4$$

In contribution 2, we claim $1 + q \ll \frac{d_{\text{full}}}{d_{\text{quant}}}$ because d_{full} is unbounded as T goes to infinity. In the real world example, $\frac{d_{\text{full}}}{d_{\text{quant}}} = 4$ is still much greater than $1 + q$. Furthermore, since the local computation is much cheaper than data communication, we conclude that the benefit from less communication by quantization (4 times less bits) overwhelm the slowdown effect from quantization ($(1 + q)$ times more local computation).

6 Experiments

In this section, we provide experimental results of FedAQ in homogeneous local data distribution settings. We compare FedAQ with other quantization-based federated optimization algorithms, FedPAQ [27] and FedCOMGATE [9]. FedAvg [26] and FedAC [43], federated optimization algorithms without quantization, are also our baselines. We empirically validate the performance of 5 algorithms on classical classification tasks on MNIST[18] and CIFAR-10[17] datasets in the distributed learning environment. We consider three objective functions i) A strongly convex objective of l_2-regularized logistic regression model on the MNIST dataset, ii) A non convex objective of training a multilayer perceptron on the MNIST data, and iii) A non convex objective of training a convolution neural network (CNN) on the CIFAR-10 dataset.

6.1 Experimental Setup

Implementation Environment. We follow the implementation setup in [9]. We use the Distributed library of PyTorch to implement our algorithm because this library allows us to simulate real-world communication and distributed training. The 18 cores of Intel Xeon E5-2676 CPU are used as computing sources. Each core is considered as one local client. We use 16 cores for strongly convex MNIST, 18 cores for the non-convex MNIST, and 8 cores for the CIFAR-10. For MNIST, the strongly convex experiment and the non-convex one respectively run for 300 rounds of communication with 20 local updates and 50 rounds of communication with 100 local updates. The CIFAR-10 experiment runs for 100 rounds of communication with 100 local updates.

Datasets. For image classification tasks, we choose two main classical image datasets: MNIST and CIFAR-10. Since we assume homogeneous settings, data is distributed homogeneously among clients, which also means each device has access to all 10 classes.

Hyperparameter Choice. The important hyperparmeters in our experiments are learning rates for each algorithm. For the client learning rate η, we respectively use 0.002, 0.1, and 0.01 for strongly convex MNIST, non-convex MNIST, and CIFAR-10 for all algorithms. For FedAQ and FedAC, once we set the value of μ, other hyperparameters (γ, α, β) are automatically determined (See condition set (33) and (34)). Thus, we choose 0.1, 0.01, and 0.2 for μ value for strongly convex MNIST, non-convex MNIST, and CIFAR-10. Since too large μ leads to slow convergence and too small μ leads to unstable training, we get these μ values by tuning μ appropriately. FedCOMGATE has a server learning rate, and we set this value as 1 for all experiments.

Quantization Bits. We have three quantization-based federated algorithms: FedAQ, FedPAQ, FedCOMGATE. We quantize the updates from 32 bits to 8 bits for all quantization-based algorithms in both MNIST and CIFAR-10. Additionally, particularly for FedAQ in non-convex experiments, we consider 4 bits quantization as well. Since FedAQ sends twice as many messages as FedPAQ or FedCOMGATE at every synchronization when we use 8 bits quantization for all cases, we apply 4 bits quantization to FedAQ to let FedAQ send the same amount of information in each communication round as other quantization-based algorithms for a fair comparison.

New Time Metric. In our experiments, communication between CPU cores is very fast, so it is hard to say that the environment of our experiments fully reflects the real-world federated learning when there is a heavy communication burden. Thus, we use a linear model to estimate the execution time $T_{\text{round}}(\mathcal{A})$ between two

consecutive communication rounds for real federated learning scenarios [35].

$$T_{\text{round}}(\mathcal{A}) = T_{\text{comm}}(\mathcal{A}) + T_{\text{comp}}(\mathcal{A}), \qquad T_{\text{comm}}(\mathcal{A}) = \frac{S_{\text{down}(\mathcal{A})}}{B_{\text{down}}} + \frac{S_{\text{up}(\mathcal{A})}}{B_{\text{up}}}$$
$$T_{\text{comp}}(\mathcal{A}) = \max_j T^j_{\text{client}}(\mathcal{A}) + T_{\text{server}}(\mathcal{A}), \qquad T^j_{\text{client}}(\mathcal{A}) = R_{\text{comp}} T^j_{\text{sim}}(\mathcal{A}) + C_{\text{comp}}$$

Since $T_{\text{server}}(\mathcal{A})$ is relatively smaller than $T^j_{\text{client}}(\mathcal{A})$, we ignore $T_{\text{server}}(\mathcal{A})$ in our experiments. We get client download size $S_{\text{down}(\mathcal{A})}$ and upload size $S_{\text{up}(\mathcal{A})}$ from the number of neural network parameters. $\max_j T^j_{\text{sim}}(\mathcal{A})$ is the computation time in our simulation.

$$B_{\text{down}} \sim 0.75\text{MB/secs},\ B_{\text{up}} \sim 0.25\text{B/secs},\ R_{\text{comp}} \sim 7,\ C_{\text{comp}} \sim 10\text{secs}$$

[35] estimate each value of the above parameters from a real world cross-device FL system. The upload bandwidth B_{up} is generally smaller than download bandwidth B_{down}. We define human time as the parallel time estimated by this new time metric.

6.1.1 Training Models

For MNIST, we use a l_2-regularized logistic regression model for the strongly convex case and a multilayer perceptron (MLP) with two hidden layers for the non-convex case. For CIFAR-10, we use a Convolutional Neural Network (CNN). Here, we note that the number of parameters in a neural network model is directly related to the number of communicated bits. We discuss more details as follows.

MLP Model for MNIST. We use a multilayer perceptron (MLP) with two hidden layers. Each hidden layer consists of 200 neurons with ReLU activations. Thus, we compute the total number of parameters in this MLP model as below.

$$\begin{aligned}(\#\text{ of MLP parameters}) &= (\#\text{ of input features}) \times (\#\text{ of neurons in the 1st layer}) \\ &+ (\#\text{ of neurons in the 1st layer}) \times (\#\text{ of neurons in the 2nd layer}) \\ &+ (\#\text{ of neurons in the 2nd layer}) \times (\#\text{ of MNIST classes}) \\ &+ (\#\text{ of neurons in the 1st layer}) + (\#\text{ of neurons in the 2nd layer}) \\ &+ (\#\text{ of MNIST classes}) \\ &= 28 \times 28 \times 200 + 200 \times 200 + 200 \times 10 + 200 + 200 + 10 = 199210\end{aligned}$$

Finally, we derive $S_{\text{up}}(\mathcal{A})(= S_{\text{down}}(\mathcal{A}))$, defined in section 6.1 (New time metric), by using the above fact. We use 32 bits floating-point if there is no quantization.

$$\begin{aligned}S_{\text{up}}(\mathcal{A}) &= (\#\text{ of device}) \times (\#\text{ of MLP parameters}) \times (\#\text{ of bits}) \\ &= 18 \times 199210 \times 32 = 114744960\end{aligned}$$

The FedAvg algorithm follows the above calculation. If we use 8 bits quantization for FedPAQ, FedCOMGATE, and FedAQ, (# of bits) in the above equation will respectively be 8, 8, and 16. Since FedAQ sends twice as many messages as others at every communication round, (# of bits) for FedAQ is 16. Similarly, (# of bits) for FedAC, which has no quantization, is 64.

CNN Model for CIFAR-10. We use a CNN model, which consists of two 2-dimensional convolutional layers, two max pooling layers, and two fully connected layers. The ReLU activations are used in this CNN model. Let's clarify (# of input channel, # of output channel, kernel size, stride) for convolutional layers. We respectively use (3, 20, 5, 1), (20, 50, 5, 1) for the 1st and 2nd convolutional layer. Let's denote each convolutional layer and fully connected layer as CONV1, CONV2, FC3, FC4. At first, the activation shape of input layer for CIFAR-10 is (32, 32, 3). Then, we get the activation shape after CONV1 and the number of parameters for CONV1.

$$\begin{aligned}(\text{width of activation shape}) &= \frac{(\text{width of previous activation shape}) - \text{kernel size} + 1}{\text{stride}} \\ &= \frac{32-5+1}{1} = 28 \Rightarrow \text{activation shape} = (28, 28, 20) \\ (\#\text{ of CONV1 parameters}) &= \Big(\text{kernel size} \times \text{kernel size} \\ &\quad \times (\#\text{ of filters in the previous layer}) + 1\Big) \\ &\quad \times (\#\text{ of filters in the current layer}) \\ &= (5 \times 5 \times 3 + 1) \times 20 = 1520\end{aligned}$$

The activation shape becomes (14, 14, 20) after max pooling. There are no learnable parameters in pooling layers. We do similar calculation for CONV2.

$$\begin{aligned}(\text{width of activation shape}) &= \frac{(\text{width of previous activation shape}) - \text{kernel size} + 1}{\text{stride}} \\ &= \frac{14-5+1}{1} = 10 \Rightarrow \text{activation shape} = (10, 10, 50) \\ (\#\text{ of CONV2 parameters}) &= \Big(\text{kernel size} \times \text{kernel size} \times (\#\text{ of filters in the previous layer}) \\ &\quad + 1\Big) \times (\#\text{ of filters in the current layer}) \\ &= (5 \times 5 \times 20 + 1) \times 50 = 25050\end{aligned}$$

The activation shape becomes (5, 5, 50) after second max pooling. Then, we calculate the number of parameters in FC3 and FC4 similar to the MLP case.

$$\begin{aligned}(\#\text{ of FC3 parameters }) &= (5 \times 5 \times 50) \times 512 + 512 = 640512 \\ (\#\text{ of FC4 parameters }) &= 512 \times 10 + 10 = 5130\end{aligned}$$

Thus, the total number of parameters in this CNN model is

$$\begin{aligned}(\#\text{ of CNN parameters}) &= (\#\text{ of CONV1 parameters}) + (\#\text{ of CONV2 parameters}) \\ &\quad + (\#\text{ of FC3 parameters}) + (\#\text{ of FC4 parameters}) \\ &= 1520 + 25050 + 640512 + 5130 = 672212\end{aligned}$$

Finally, we derive $S_{\text{up}}(\mathcal{A})(= S_{\text{down}}(\mathcal{A}))$ in this case.

$$\begin{aligned}S_{\text{up}}(\mathcal{A}) &= (\#\text{ of device}) \times (\#\text{ of CNN parameters}) \times (\#\text{ of bits}) \\ &= 8 \times 672212 \times 32 = 172086272\end{aligned}$$

We can do the similar discussion in the MLP case when it comes to applying this to quantization-based federated optimization algorithms.

6.2 Experimental Results

In our experiments on both MNIST and CIFAR-10, we verify how the global training loss and test accuracy of five algorithms change with respect to communication rounds, the number of bits communicated between one client and the server during the uplink, and human time. We provide both qualitative analysis and quantitative results for plots.

6.2.1 Qualitative Analysis

Strongly Convex Case. In this experiment, we compare FedAQ under the condition set (33) and set (34) with FedAvg, FedPAQ, FedCOMGATE, and FedAC-I. We denote each FedAQ as FedAQ-I and FedAQ-II. As we observe the theoretical benefits of FedAQ over other methods in section 5, FedAQ-I outperforms all other quantization-based federated optimization algorithms and FedAC-I in all plots (See each first row of Figure 1, 2). However, although FedAQ-II shows the fast convergence speed, the training process is unstable. Thus, we only use FedAQ-I for further non-convex experiments. FedAC and FedAQ in non-convex experiments indicate FedAC-I and FedAQ-I.

Non-Convex Case. Each second row of Figure 1, 2, and Figure 3 clearly demonstrates that FedAQ with 4 bits quantization outperforms other algorithms in all plots. In terms of communication rounds, accelerated algorithms, FedAQ and FedAC, converge faster than other algorithms. We also observe that quantization does not lead to slower convergence, which means we can apply an efficient quantization scheme to make communication efficient FL systems without sacrificing convergence speed. The plots related to communicated bits are helpful to interpret how algorithms work well in situations with heavy communication. FedAQ with 8 bits quantization shows comparable performance relative to FedPAQ and FedCOMGATE with the help of acceleration, even though FedAQ sends more updates during every synchronization. When we use 4 bits quantization for FedAQ to make the number of communicated bits the same for all quantization-based algorithms during synchronization, FedAQ shows a much faster convergence speed with regard to the number of communicated bits. However, plots of communicated bits fail to reflect how algorithms converge in real estimated time for FL scenarios, which consists of both communication and computation. Thus, we further analyze algorithms with human time. We observe that FedAQ with 8 quantization bits performs slightly better than FedPAQ and FedCOMGATE for both MNIST and CIFAR-10. This occurs because while all quantization-based algorithms send the same number of communicated bits, the number of communication rounds for FedAQ is much smaller than others. Then, this also indicates that FedAQ takes less computation time than other methods while reaching the same accuracy.

6.2.2 Quantitative Results

We provide quantitative results to help readers understand plots better. To be specific, for all plots, we observe the number of communication rounds, the number of communicated bits, and the human time required to achieve a particular test accuracy by each federated optimization algorithm.

For the strongly convex experiment on MNIST (See the first row of Figure 2), the number of communication rounds required to achieve 90.28% test accuracy by FedAvg, FedPAQ(8bits), FedCOMGATE(8bits), FedAC-I, FedAQ-I(8bits), FedAQ-II(8bits) are respectively 217, 216, 260, 28, 26, 99. The number of communicated bits required to achieve the same accuracy are respectively 5.4e7, 1.4e7, 1.6e7, 1.4e7, 3.3e6, 1.2e7. Lastly, the required human time are respectively 3220s, 2760s, 3336s, 484s, 344s, 1323s. In this experiment, FedAQ-I(8bits) requires the smallest number of communication rounds, the smallest number of communicated bits, and the shortest human time to achieve the same test accuracy. These experimental results support the validity of our theoretical analysis on strongly convex cases.

For the non-convex experiment on MNIST (See the second row of Figure 2), the number of communication rounds required to achieve 97.6% test accuracy by FedAvg, FedPAQ(8bits), FedCOMGATE(8bits), FedAC,

FedAQ(8bits), FedAQ(4bits) are respectively 23, 48, 38, 18, 18, 16. The number of communicated bits required to achieve the same accuracy are respectively 1.5e8, 7.6e7, 6.1e7, 2.3e8, 5.7e7, 2.5e7. Finally, the required human time are respectively 2424s, 2311s, 1834s, 3327s, 1248s, 805s. Thus, we conclude that FedAQ(4bits) outperforms other algorithms, and even FedAQ(8bits) needs smaller number of communicated bits/less human time to achieve the goal accuracy than FedPAQ(8bits)/FedCOMGATE(8bits).

For the non-convex experiment on CIFAR-10 (See Figure 3), the number of communication rounds required to achieve 65.4% test accuracy by FedAvg, FedPAQ(8bits), FedCOMGATE(8bits), FedAC, FedAQ(8bits), FedAQ(4bits) are respectively 98, 89, 95, 49, 50, 48. The number of communicated bits required to achieve the same accuracy are respectively 2.1e9, 4.8e8, 5.1e8, 2.1e9, 5.4e8, 2.6e8. Finally, the required human time are respectively 31798s, 11526s, 12240s, 28720s, 9902s, 6464s. As with the non-convex experiment on MNIST, FedAQ(4bits) outperforms other algorithms, and even FedAQ(8bits) requires less human time to achieve the same accuracy than FedPAQ(8bits)/FedCOMGATE(8bits).

Remark 24: Our current experimental setup only allows us to scale the number of clients up to the number of CPU cores in our machine. Since FedAQ achieves linear speed up in the number of workers with much fewer communication rounds than other quantization based methods, we expect FedAQ to outperform other methods by an even larger margin as we scale the number of workers.

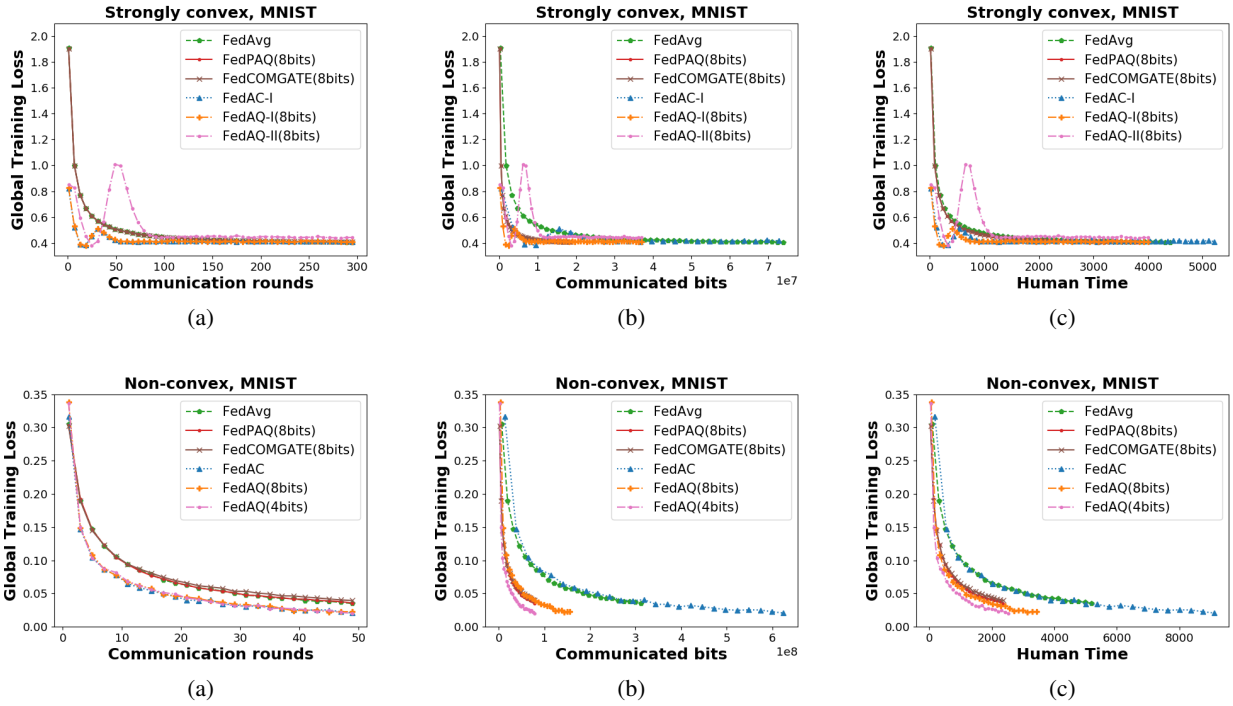


Figure 1: Comparing FedAQ with FedAvg, FedPAQ, FedCOMGATE, and FedAC on MNIST with Strongly Convex Settings (first row) and Non-Convex Settings (second row). We observe how the global training loss changes across communication rounds (first column), communicated bits (second column), and human time (third column). FedAQ-I(8bits) and FedAQ(4bits) respectively outperform other algorithms for strongly convex settings and non-convex settings. FedAQ(4bits) sends the same number of communicated bits as FedPAQ(8bits) and FedCOMGATE(8bits) in each communication round, which indicates a fair comparison (See Quantization bits in section 6.1).

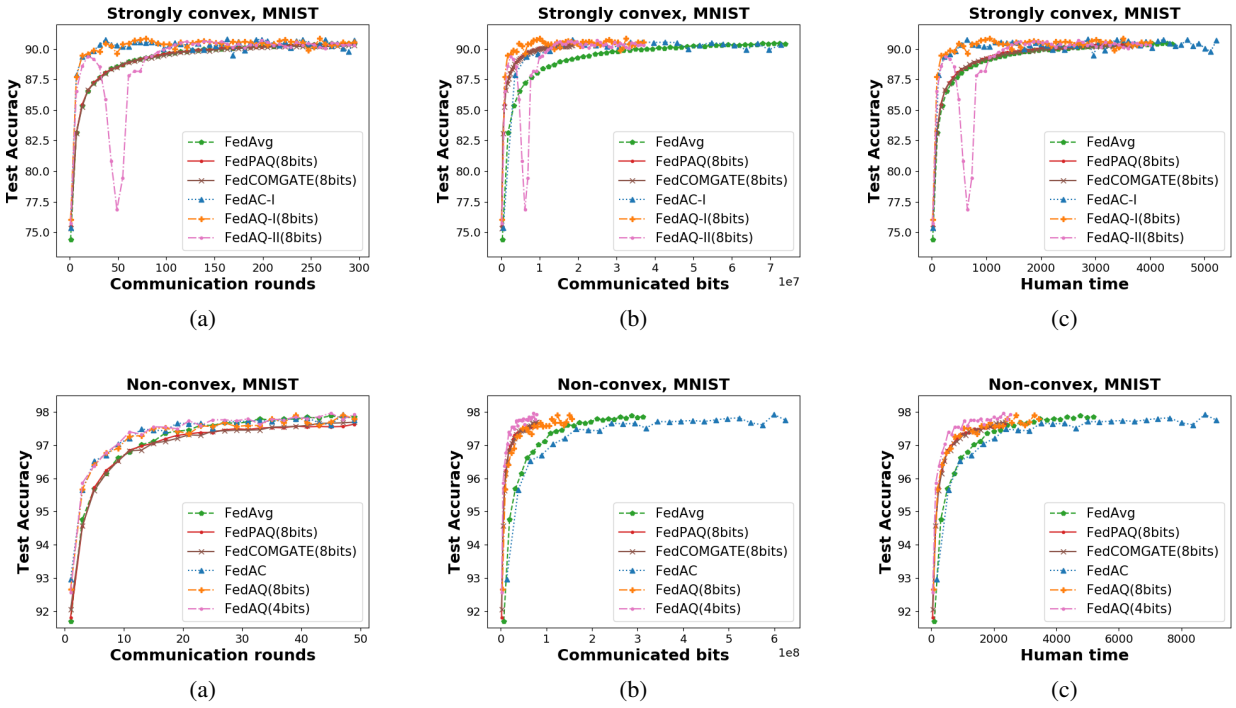


Figure 2: Comparing FedAQ with FedAvg, FedPAQ, FedCOMGATE, and FedAC on MNIST with Strongly Convex Settings (first row) and Non-Convex Settings (second row). We observe how the test accuracy changes across communication rounds (first column), communicated bits (second column), and human time (third column). FedAQ-I outperforms other algorithms in all plots for strongly convex settings. Moreover, FedAQ(4bits) outperforms other algorithms in all plots for non-convex settings.

7 Conclusion

To sum up, we propose a novel communication-efficient federated optimization algorithm, FedAQ, that successfully incorporates accelerated multiple local updates and quantization with solid theoretical guarantees in strongly-convex and homogeneous settings. In the future, further theoretical guarantees of FedAQ on convex and non-convex functions should be discussed. Also, the convergence analysis of FedAQ on heterogeneous settings can be an interesting topic. Even though Federated Learning systems provide some level of privacy to the clients as their explicit data is not shared with the servers, careful examination of FL systems including FedAQ is necessary to examine how much privacy do they actually provide as information is shared in form of the iterates.

References

[1] D. Alistarh, D. Grubic, J. Li, R. Tomioka, and M. Vojnovic. Qsgd: Communication-efficient sgd via gradient quantization and encoding. Advances in Neural Information Processing Systems, 30:1709–1720, 2017.

[2] N. Bansal and A. Gupta. Potential-function proofs for gradient methods. Theory of Computing, 15(1):1–32, 2019.

[3] D. Basu, D. Data, C. Karakus, and S. Diggavi. Qsparse-local-sgd: Distributed sgd with quantization, sparsification, and local computations. arXiv preprint arXiv:1906.02367, 2019.

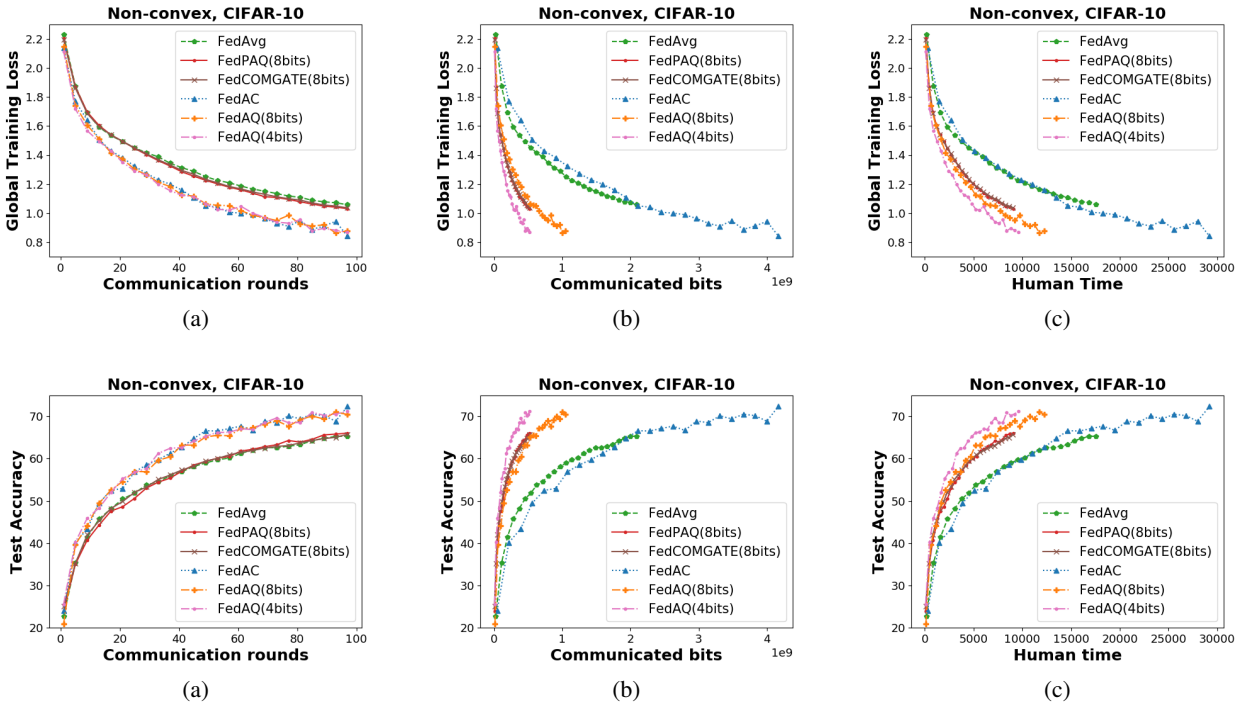


Figure 3: Comparing FedAQ with FedAvg, FedPAQ, FedCOMGATE, and FedAC on CIFAR-10. We observe how the global training loss and test accuracy change across communication rounds (first column), communicated bits (second column), and human time (third column). We use a CNN model for CIFAR-10. Similar to the MNIST experiment, FedAQ (4 bits) outperforms all other algorithms in every case.

[4] J. Bernstein, Y.-X. Wang, K. Azizzadenesheli, and A. Anandkumar. signsgd: Compressed optimisation for non-convex problems. In International Conference on Machine Learning, pages 560–569. PMLR, 2018.

[5] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. Advances in neural information processing systems, 33:1877–1901, 2020.

[6] S. Ghadimi and G. Lan. Optimal stochastic approximation algorithms for strongly convex stochastic composite optimization i: A generic algorithmic framework. SIAM Journal on Optimization, 22(4):1469–1492, 2012.

[7] F. Haddadpour, M. M. Kamani, M. Mahdavi, and V. Cadambe. Local sgd with periodic averaging: Tighter analysis and adaptive synchronization. In Advances in Neural Information Processing Systems, pages 11082–11094, 2019.

[8] F. Haddadpour, M. M. Kamani, M. Mahdavi, and V. Cadambe. Trading redundancy for communication: Speeding up distributed sgd for non-convex optimization. In International Conference on Machine Learning, pages 2545–2554. PMLR, 2019.

[9] F. Haddadpour, M. M. Kamani, A. Mokhtari, and M. Mahdavi. Federated learning with compression: Unified analysis and sharp guarantees. In International Conference on Artificial Intelligence and Statistics, pages 2350–2358. PMLR, 2021.

[10] F. Haddadpour and M. Mahdavi. On the convergence of local descent methods in federated learning. arXiv preprint arXiv:1910.14425, 2019.

[11] S. Horvath, C.-Y. Ho, L. Horvath, A. N. Sahu, M. Canini, and P. Richtárik. Natural compression for distributed deep learning. arXiv preprint arXiv:1905.10988, 2019.

[12] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, et al. Advances and open problems in federated learning. arXiv preprint arXiv:1912.04977, 2019.

[13] S. P. Karimireddy, M. Jaggi, S. Kale, M. Mohri, S. J. Reddi, S. U. Stich, and A. T. Suresh. Mime: Mimicking centralized stochastic algorithms in federated learning. arXiv preprint arXiv:2008.03606, 2020.

[14] S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and A. T. Suresh. Scaffold: Stochastic controlled averaging for federated learning. In International Conference on Machine Learning, pages 5132–5143. PMLR, 2020.

[15] A. Khaled, K. Mishchenko, and P. Richtárik. Tighter theory for local sgd on identical and heterogeneous data. In International Conference on Artificial Intelligence and Statistics, pages 4519–4529. PMLR, 2020.

[16] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon. Federated learning: Strategies for improving communication efficiency. arXiv preprint arXiv:1610.05492, 2016.

[17] A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images. Manuscript, 2009.

[18] Y. LeCun. The mnist database of handwritten digits. http://yann. lecun. com/exdb/mnist/, 1998.

[19] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith. Federated learning: Challenges, methods, and future directions. IEEE Signal Processing Magazine, 37(3):50–60, 2020.

[20] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith. Federated optimization in heterogeneous networks. arXiv preprint arXiv:1812.06127, 2018.

[21] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang. On the convergence of fedavg on non-iid data. arXiv preprint arXiv:1907.02189, 2019.

[22] X. Li, B. Karimi, and P. Li. On distributed adaptive optimization with gradient compression. arXiv preprint arXiv:2205.05632, 2022.

[23] Z. Li, D. Kovalev, X. Qian, and P. Richtárik. Acceleration for compressed gradient descent in distributed and federated optimization. arXiv preprint arXiv:2002.11364, 2020.

[24] Z. Li and P. Richtárik. Canita: Faster rates for distributed convex optimization with communication compression. arXiv preprint arXiv:2107.09461, 2021.

[25] T. Lin, S. U. Stich, K. K. Patel, and M. Jaggi. Don't use large mini-batches, use local sgd. arXiv preprint arXiv:1808.07217, 2018.

[26] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas. Communication-efficient learning of deep networks from decentralized data. In Artificial intelligence and statistics, pages 1273–1282. PMLR, 2017.

[27] A. Reisizadeh, A. Mokhtari, H. Hassani, A. Jadbabaie, and R. Pedarsani. Fedpaq: A communication-efficient federated learning method with periodic averaging and quantization. In International Conference on Artificial Intelligence and Statistics, pages 2021–2031. PMLR, 2020.

[28] D. Rothchild, A. Panda, E. Ullah, N. Ivkin, I. Stoica, V. Braverman, J. Gonzalez, and R. Arora. Fetchsgd: Communication-efficient federated learning with sketching. In International Conference on Machine Learning, pages 8253–8265. PMLR, 2020.

[29] N. Singh, D. Data, J. George, and S. Diggavi. Squarm-sgd: Communication-efficient momentum sgd for decentralized optimization. IEEE Journal on Selected Areas in Information Theory, 2021.

[30] S. U. Stich. Local sgd converges fast and communicates little. arXiv preprint arXiv:1805.09767, 2018.

[31] S. U. Stich and S. P. Karimireddy. The error-feedback framework: Better rates for sgd with delayed gradients and compressed communication. arXiv preprint arXiv:1909.05350, 2019.

[32] A. T. Suresh, X. Y. Felix, S. Kumar, and H. B. McMahan. Distributed mean estimation with limited communication. In International Conference on Machine Learning, pages 3329–3337. PMLR, 2017.

[33] T. Vogels, S. P. Karinireddy, and M. Jaggi. Powersgd: Practical low-rank gradient compression for distributed optimization. Advances In Neural Information Processing Systems 32 (Nips 2019), 32(CONF), 2019.

[34] H. Wang, S. Sievert, Z. Charles, S. Liu, S. Wright, and D. Papailiopoulos. Atomo: Communication-efficient learning via atomic sparsification. arXiv preprint arXiv:1806.04090, 2018.

[35] J. Wang, Z. Charles, Z. Xu, G. Joshi, H. B. McMahan, M. Al-Shedivat, G. Andrew, S. Avestimehr, K. Daly, D. Data, et al. A field guide to federated optimization. arXiv preprint arXiv:2107.06917, 2021.

[36] J. Wang and G. Joshi. Cooperative sgd: A unified framework for the design and analysis of communication-efficient sgd algorithms. arXiv preprint arXiv:1808.07576, 2018.

[37] J. Wang, Z. Xu, Z. Garrett, Z. Charles, L. Liu, and G. Joshi. Local adaptivity in federated learning: Convergence and consistency. arXiv preprint arXiv:2106.02305, 2021.

[38] Y. Wang, L. Lin, and J. Chen. Communication-efficient adaptive federated learning. arXiv preprint arXiv:2205.02719, 2022.

[39] J. Wangni, J. Wang, J. Liu, and T. Zhang. Gradient sparsification for communication-efficient distributed optimization. arXiv preprint arXiv:1710.09854, 2017.

[40] B. Woodworth, K. K. Patel, S. Stich, Z. Dai, B. Bullins, B. Mcmahan, O. Shamir, and N. Srebro. Is local sgd better than minibatch sgd? In International Conference on Machine Learning, pages 10334–10343. PMLR, 2020.

[41] H. Yu, R. Jin, and S. Yang. On the linear speedup analysis of communication efficient momentum sgd for distributed non-convex optimization. In International Conference on Machine Learning, pages 7184–7193. PMLR, 2019.

[42] H. Yu, S. Yang, and S. Zhu. Parallel restarted sgd with faster convergence and less communication: Demystifying why model averaging works for deep learning. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 33, pages 5693–5700, 2019.

[43] H. Yuan and T. Ma. Federated accelerated stochastic gradient descent. Advances in Neural Information Processing Systems, 33, 2020.

Federated Truth Discovery for Mobile Crowdsensing with Privacy-Preserving Trustworthiness Assessment

Leye Wang[1,2], Guanghong Fan[1,2], Xiao Han[3,*]
[1]Key Lab of High Confidence Software Technologies (Peking University), Ministry of Education, China
[2]School of Computer Science, Peking University, China
[3]School of Information Management and Engineering, Shanghai University of Finance and Economics, China
leyewang@pku.edu.cn, fgh@stu.pku.edu.cn, xiaohan@mail.shufe.edu.cn

Abstract

With the prevalence of smart mobile devices empowered by considerable sensing capabilities, crowdsensing has become one promising way to sense urban phenomena (e.g., traffic and environment) at a large scale. In crowdsensing, a fundamental issue is discovering the truth from participants' noisy sensed data. Traditionally, participants need to upload their raw sensed data with locations for truth discovery, but this may leak participants' private information such as home and work locations. In this paper, we propose a federated truth discovery method that can learn the truth without collecting participants' sensed data and locations. Our method ensures that the obtained truth quality has no performance loss compared to the original truth discovery method if all the participants keep online; even if some participants lose connections unpredictably, our method can still learn the truth based on rest participants' data. Meanwhile, as participants' sensed data are unknown to the server, it is hard for the crowdsensing organizer to justify each participant's sensing trustworthiness. This brings difficulties to crowdsensing management such as participant recruitment and incentive allocation. We further propose a federated ranking mechanism to generate a leader-board for participants' trustworthiness, which can also tolerate participants' connection loss. Both theoretical analysis and real-data empirical evaluations have been done to verify the effectiveness of FedTruthFinder.

1 Introduction

With the popularity of smart mobile devices, such as smartphones, pads, and vehicles, crowdsensing has become one promising paradigm for sensing urban dynamics [3]. A typical crowdsensing process first recruits participants and then asks them to upload the data of interest to the central server. Afterward, the server aggregates the data from participants toward a synthetic sensed result [39, 11]. While each participant's sensed data may include noise, an important issue is how to ensure the accuracy of the aggregated sensed result, often called *truth discovery* [26, 12, 24].

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

*Corresponding author

Many research studies have been devoted to the truth discovery for crowdsensing applications. Generally, most relevant studies model participant trustworthiness and sensed data confidence iteratively to obtain the final aggregated sensed result [36]. The basic idea is that a high-credit participant's sensed data should be assigned with high confidence; a user whose sensed data are more confident should be more trustworthy. While this has been verified to be effective, the iterative computation process needs a central server to collect every participant's raw sensed data, which may bring privacy threats to crowdsensing participants. For example, crowdsensing usually asks participants to do sensing tasks at specific locations, and thus most sensed data are associated with detailed location information [39]. The traditional truth discovery process would inevitably leak crowdsensing participants' visited locations during sensing periods, which may be exploited by malicious third parties to conduct serious location privacy breaches such as physical stalking [19]. With the user privacy and personal data regularization (e.g., General Data Protection Regulation[1]) becoming more and more important nowadays, we believe that a privacy-preserving truth discovery algorithm is urgently required for facilitating more extensive crowdsensing campaigns in practice.

Privacy-preserving truth discovery has been recently studied in crowdsensing. Existing studies are usually based on some hardly realized assumptions such as every participant being always online and/or non-colluding [13, 15, 14][2], or two non-colluding servers (or fog nodes) are required [23, 40, 38, 37]. These assumptions hinder the practical applications of the prior mechanisms. More importantly, almost all the prior studies do not discuss two fundamental issues in privacy-preserving truth discovery.

i) **Participants' Completed Tasks Protection.** Existing mechanisms focus on protecting participants' sensed data, but most assume that participants' completed tasks are known to the crowdsensing server [34, 13, 14, 40]. However, sometimes task information is more sensitive than sensed data. Suppose the tasks are air quality sensing at certain points of interest. If a participant's task completion information is disclosed (e.g., finish a sensing task at the Times Square NYC), then her location is revealed without the need to know her sensed air quality value.

ii) **Participants' Trustworthiness Assessment.** Trustworthiness assessment is a key part of crowdsensing for participant recruitment and incentive design [17], while privacy-preserving truth discovery needs to hide participants' trustworthiness scores for privacy protection. To address the dilemma, a privacy-preserving trustworthiness assessment method needs to be proposed.

In this paper, we aim to design a novel and practical privacy-preserving truth discovery mechanism for crowdsensing to overcome the pitfalls of prior studies. In particular, our design follows the federated learning (FL) paradigm [5, 35]. We thus call our method as *FedTruthFinder*. In general, FL requires user clients to do some local computation (e.g., learning the gradients for updating the parameters of the model) on their devices and then only upload the computation results instead of raw data. For a specific algorithm, the local computation and uploading process usually incorporates certain secure mechanisms (e.g., homomorphic encryption and secure multi-party computation) to ensure that no user privacy is leaked theoretically [2, 10]. Based on FL, we aim to consider the following specific issues in the design of FedTruthFinder.

No Accuracy Loss: We expect that FedTruthFinder will not hurt the accuracy of the aggregated sensed results compared to the centralized truth discovery. Without the loss of accuracy, crowdsensing organizers would be likely to adopt the method in practice.

No Third Party: Crowdsensing involves a central server and a set of participants' clients. To make FedTruthFinder easy to deploy, we do not want to introduce any more third parties which are usually hard to find in reality [1, 27].

Robustness against Unpredictable Connection Loss: While crowdsensing participants travel around the whole city, their device connection with the central server is not always stable. Hence, it is necessary to make FedTruthFinder effective when some participants lose connections suddenly.

[1] `https://gdpr.eu/eu-gdpr-personal-data`

[2] Some participants' relations can be very close such as family members, and thus it is non-reasonable to assume that they will not collude with each other.

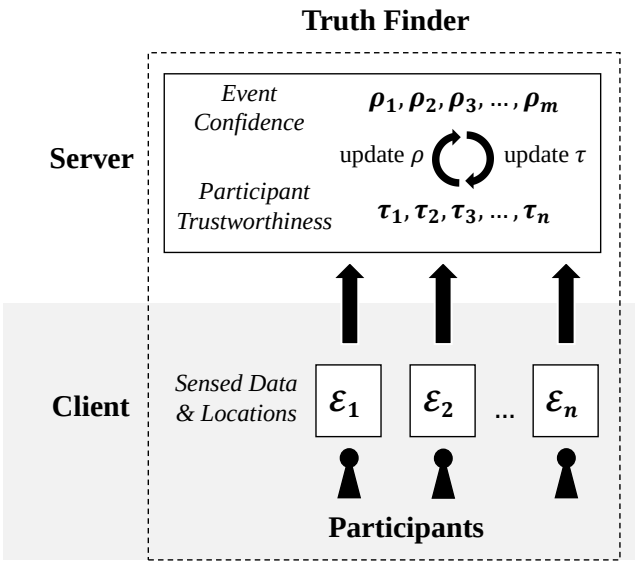


Figure 1: Overview of Iterative Truth Discovery

With the above issues in consideration, this paper makes the following contributions:

(1) To the best of our knowledge, this paper is the first study that addresses the problem of privacy-preserving crowdsensing truth discovery with (i) *participants' completed task protection*, and (ii) *participants' trustworthiness assessment*.

(2) We propose *FedTruthFinder*, a novel privacy-preserving truth discovery mechanism following the federated learning paradigm [35]. FedTruthFinder does not need any third party and can tolerate participants' unpredictable connection loss. The two key components of FedTruthFinder are (i) *federated confidence computation* to learn the probability of a sensed event, and (ii) *federated trustworthiness ranking* to assess participants' sensed data quality.

(3) We have conducted both theoretical analysis and empirical evaluations for FedTruthFinder. In particular, FedTruthFinder can reduce the system failure probability significantly compared to state-of-the-art privacy-preserving truth discovery approaches [1, 34], and achieve good detection accuracy.

2 Preliminary: Truth Discovery

Truth discovery algorithms usually follow an iterative method to calibrate user trustworthiness and data confidence alternatively until convergence [36]. Figure 1 shows the framework of iterative truth discovery methods. In this paper, for clarity, we assume that sensed data is a binary spatial event. That is, for a specific location, the sensed data can be 1 or 0. Our method can be easily extended to multi-class and continuous-value events (see Appendix).

As shown in Figure 1, first, participants upload all of their sensed data and locations $\mathcal{E}_i$ to the central server. The central server would assign an initial trustworthiness score τ_i to each participant u_i (e.g., 0.9 by assuming that 90% of the sensed data are accurate). Then, for each sensed event e_j, the truth discovery algorithm will calculate its confidence ρ_j (i.e., the probability of $e_j = 1$) by considering the users who have sensed e_j as:

$$\rho_j = F_\rho(\mathcal{U}_{j,1}, \mathcal{U}_{j,0}) \tag{72}$$

where $\mathcal{U}_{j,k}$ is the users who have sensed the event e_j with the reported data k; F_ρ is an event confidence calculation function which we will elaborate on later.

With ρ_j for each event e_j, we can then update the trustworthiness score τ_i of each participant u_i by:

$$\tau_i = F_\tau(\mathcal{E}_{i,1}, \mathcal{E}_{i,0}) \tag{73}$$

where $\mathcal{E}_{i,k}$ is the users' sensed event set with the reported data k; F_τ is a user trustworthiness calculation function which we will elaborate on later.

Once τ_i is updated for each user u_i, we can continue updating ρ_j for each event e_j according to Eq. 72, and so on, leading to an alternative updating process for both τ_i and ρ_j. This process can be terminated after a fixed number of iterations or until convergence. Next, we elaborate on the common choices of F_ρ and F_τ in literature.

Sum Function

An intuitive selection of the updating functions of F_ρ and F_τ is the weighted sum:

$$\rho_j = F_\rho(\mathcal{U}_{j,1}, \mathcal{U}_{j,0}) = \frac{\sum_{u_i \in \mathcal{U}_{j,1}} \tau_i}{\sum_{u_i \in \mathcal{U}_{j,1}} \tau_i + \sum_{u_k \in \mathcal{U}_{j,0}} \tau_k} \tag{74}$$

$$\tau_i = F_\tau(\mathcal{E}_{i,1}, \mathcal{E}_{i,0}) = \frac{\sum_{e_j \in \mathcal{E}_{i,1}} \rho_j + \sum_{e_k \in \mathcal{E}_{i,0}} 1 - \rho_k}{|\mathcal{E}_{i,1}| + |\mathcal{E}_{i,0}|} \tag{75}$$

Logistic Function

Another widely used updating function is the Logistic function [36]. Its basic idea is seeing every user independently, so that the probability of event happening, i.e., $e_j = 1$, can be formulated as:

$$\rho_j = 1 - \prod_{u_i \in \mathcal{U}_{j,1}} (1 - \tau_i) \tag{76}$$

As $1 - \tau_i$ may often be small and multiplying many of them may lead to underflow, prior studies proposed to use the logarithm to define a log-trustworthiness score of u_i as [36]:

$$\tau_i^* = -\ln(1 - \tau_i) \tag{77}$$

Similarly, a log-confidence score of event e_j is defined as:

$$\rho_j^* = -\ln(1 - \rho_i) \tag{78}$$

Then, we can infer

$$\rho_j^* = \sum_{u_i \in \mathcal{U}_{j,1}} \tau_i^* \tag{79}$$

The above equation does not consider the users' trustworthiness who report $e_j = 0$, and thus we refine it:

$$\rho_j^* = \sum_{u_i \in \mathcal{U}_{j,1}} \tau_i^* - \sum_{u_k \in \mathcal{U}_{j,0}} \tau_k^* \tag{80}$$

Finally, a logistic function is used to calculate the final confidence ρ_j of event e_j [36]:

$$\rho_j = F_\rho(\mathcal{U}_{j,1}, \mathcal{U}_{j,0}) = (1 + e^{-\rho_j^*})^{-1} \tag{81}$$

τ_i is updated same as Eq. 75.

3 Federated Truth Discovery: Overview and Key Issues

3.1 Overall Design

Figure 2 overviews the workflow of our method *FedTruthFinder*. The general design principle follows the federated learning paradigm [35], which requires the user clients to conduct local computations of their raw

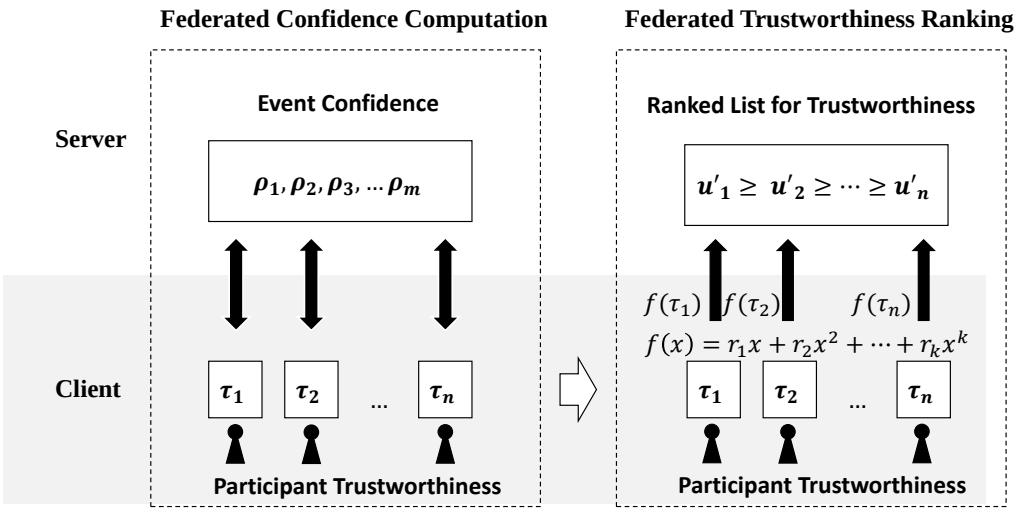


Figure 2: Overview of FedTruthFinder

data and then upload processed data that do not reveal the user's privacy to the server. With these uploaded privacy-preserving data, the server can still find the aggregate truth of the sensed events, the same as the server receiving the raw sensed data and locations from participants.

By analyzing the original truth discovery algorithm in the previous section, we note that there exist two alternative computation processes: (i) *ρ-computation*: updating the confidence ρ_j for each event e_j, and (ii) *τ-computation*: updating the trustworthiness τ_i for each participant u_i. In the two computation processes, τ-computation (e.g., Eq. 75) can be naturally offloaded to each participant u_i's device, as long as the server sends all the current $\rho_j, \forall e_j$ to the participants. However, ρ-computation needs to know each user's sensed data (and locations) and then do aggregation (e.g., sum). This needs a dedicated design to enable the privacy-preserving truth discovery, which will be illustrated in Sec. 4.

Besides, the trustworthiness of each participant's sensed data (i.e., τ_i) is a key metric in crowdsensing organization for participant recruitment and incentive allocation. Hence, we also design a federated privacy-preserving mechanism to rank participants' trustworthiness. Particularly, instead of transferring raw τ_i to the server, we leverage certain security mechanisms to upload $f(\tau_i)$ to the server, while $f(\tau_i)$ keeps the same ranking orders as τ_i. Particularly, in FedTruthFinder, $f(\tau_i) = r_1\tau_i + r_2\tau_i^2 + ... + r_k\tau_i^k$, where $r_i > 0$. In this regard, even though the server cannot know the specific τ_i of each participant u_i, the ranked list of participants according to the trustworthiness can still be learned with $f(\tau_i)$. Specifically, during the whole computation process, the server cannot know r_i, and each participant will also not know all the r_i, so that none of the server or participant can infer other participants' private τ_i. How to compute $f(\tau_i)$ securely will be introduced in Sec. 5.

3.2 Key Issues

Issue 1. Privacy-Preserving ρ-computation: *Suppose there are a set of crowdsensing participants $\mathcal{U}$ and a set of spatial events to sense $\mathcal{E}$, each participant $u_i (\in \mathcal{U})$ with sensed events $\mathcal{E}_{i,1} (\subseteq \mathcal{E})$ and $\mathcal{E}_{i,0} (\subseteq \mathcal{E})$ corresponding to the sensed value being 1 and 0, respectively. How to calculate confidence ρ_j for each event $e_j (\in \mathcal{E})$ while every participant u_i will not leak $\mathcal{E}_{i,1}$, $\mathcal{E}_{i,0}$, and $\mathcal{E}_{i,1} \cup \mathcal{E}_{i,0}$ to the server and other participants?*

Some factors need to be carefully considered:

(1) **Computation**: In ρ-computation, the value to share is a complicated equation instead of a single value, and the equation may even be varied depending on the truth discovery algorithm implementation (e.g., Eq. 74 or

Eq. 81).

(2) **Network Connection**: In crowdsensing, the network connections of mobile participants may not be always stable. Hence, our mechanism should tolerate the scenario when a few participants lose connections.

(3) **No Leakage of Task Completion**: For protecting participants' privacy, not only the sensed data but also the completed tasks (i.e., $\mathcal{E}_{i,1} \cup \mathcal{E}_{i,0}$) should not be disclosed.

Issue 2. Secure Trustworthiness Ranking: *Suppose there are a set of crowdsensing participants $\mathcal{U}$ and each participant $u_i (\in \mathcal{U})$ has a private trustworthiness score τ_i. How to rank participants according to τ_i while every participant u_i will not leak τ_i to the server and other participants?*

Addressing this issue also needs to consider the unstable network connections of participant clients as aforementioned.

Remark on security definition: In this work, we assume that the crowdsensing server and participants are *semi-honest (honest-but-curious)*: they will follow our designed protocol and not maliciously modify the inputs or outputs; however, the server and the participants will try their best to infer others' data from the data that they have received. Besides, our mechanism can defend against *collusion attacks* to a certain extent (i.e., some participants may collude with each other), which we will elaborate on later.

4 Federated Truth Computation

We first introduce a basic scheme for ρ-computation in a federated manner assuming no connection loss. Then, we improve the scheme to be against the participants' unpredictable connection loss.

4.1 Basic Scheme of ρ-Computation with SSS

In this section, we propose our basic scheme for the ρ-computation problem with the federated learning paradigm leveraging secret sharing. Given a spatial crowdsensing event e_j, we first consider ρ-computation with the sum function, i.e., Eq. 74.

4.1.1 ρ-computation with the sum function.

Our process includes three steps as follows:

Step 1 (Share Dispatching). Each participant u_i dispatches d_{ij} and s_{ij} with secret sharing to all the n participants ($\mathcal{U} = \{u_1 \cdots u_n\}$), where

$$d_{ij} = \begin{cases} \tau_i & e_j \in \mathcal{E}_{i,1} \\ 0 & e_j \in \mathcal{E} \setminus \mathcal{E}_{i,1} \end{cases} \tag{82}$$

$$s_{ij} = \begin{cases} \tau_i & e_j \in \mathcal{E}_{i,1} \cup \mathcal{E}_{i,0} \\ 0 & e_j \in \mathcal{E} \setminus (\mathcal{E}_{i,1} \cup \mathcal{E}_{i,0}) \end{cases} \tag{83}$$

Specifically, d_{ij} is divided into n shares

$$\{d_{ij}^1, \cdots, d_{ij}^n\}$$

where $d_{ij}^1, \cdots, d_{ij}^{n-1}$ are random numbers and

$$d_{ij}^n = d_{ij} - \sum_{k=1}^{n-1} d_{ij}^k$$

Hence, $\sum_{k=1}^{n} d_{ij}^k = d_{ij}$. Then, u_i sends d_{ij}^k to u_k. Similarly, s_{ij} is split to n shares

$$\{s_{ij}^1, \cdots, s_{ij}^n = s_{ij} - \sum_{k=1}^{n-1} s_{ij}^k\}$$

and u_k receives s_{ij}^k from u_i.

Step 2 (Client Summation). For each event e_j, an participant u_k uploads $\hat{d}_j^k = \sum_{i=1}^{n} d_{ij}^k$ and $\hat{s}_j^k = \sum_{i=1}^{n} s_{ij}^k$ to the server.

Step 3 (Server Aggregation). After receiving $\hat{d}_j^k$ and $\hat{s}_j^k$ from $\forall u_k \in \mathcal{U}$, the server can add them together:

$$d_j = \sum_{k=1}^{n} \hat{d}_j^k = \sum_{k=1}^{n}\sum_{i=1}^{n} d_{ij}^k = \sum_{i=1}^{n}\sum_{k=1}^{n} d_{ij}^k = \sum_{i=1}^{n} d_{ij} \tag{84}$$

$$s_j = \sum_{k=1}^{n} \hat{s}_j^k = \sum_{k=1}^{n}\sum_{i=1}^{n} s_{ij}^k = \sum_{i=1}^{n}\sum_{k=1}^{n} s_{ij}^k = \sum_{i=1}^{n} s_{ij} \tag{85}$$

Then, ρ_j is computed as:

$$\rho_j = \frac{d_j}{s_j} \tag{86}$$

4.1.2 ρ-computation with the logistic function.

The computation process with the logistic function is not much different from the one with the summation function. Actually, for the event e_j, we only need to modify d_{ij} to:

$$d_{ij} = \begin{cases} -\ln(1-\tau_i) & e_j \in \mathcal{E}_{i,1} \\ \ln(1-\tau_i) & e_j \in \mathcal{E}_{i,0} \\ 0 & e_j \in \mathcal{E} \setminus (\mathcal{E}_{i,1} \cup \mathcal{E}_{i,0}) \end{cases} \tag{87}$$

Besides, we do not need s_{ij}, so every participant u_k only receives d_{ij}^k from other $u_i \in \mathcal{U}$. Finally, in Step 3, we can compute ρ_j as:

$$\rho_j = (1 + e^{-d_j})^{-1} \tag{88}$$

Remark on our novelty. In our ρ-computation for an event e_j, the participant u_i who has not sensed e_j also needs to upload data to the server, e.g., $d_{ij} = s_{ij} = 0$ for the sum function. As d_{ij} and s_{ij} are sent by secret shares, the other participants and the server would not know whether u_i senses e_j or not. In comparison, prior studies usually assume that participants send only the data of their sensed events, which may disclose user privacy from event information (e.g., event locations) [13, 15, 14, 40, 41, 38].

4.2 Connection Robustness Improvement

The basic scheme can learn ρ_j in an ideal environment when all the participants are always online. In practice, participants move around and their network connections are often sporadic. This inspires us to make three improvements to the basic scheme design.

4.2.1 Bias-avoidance adaptive truth updating.

While participants may drop during the iterative truth discovery process due to bad connections, the original event confidence updating function would be ineffective. Specifically, if u_i loses the connection at the k^{th} iteration, u_i's data would not be considered in the ρ-computation (e.g., Eq. 74) from then on. This leads to unreliable ρ_j as the finally alive participants dominate the results. To address this pitfall, we propose an adaptive updating function for k^{th} iteration's $\rho_{j,k}$ as,

$$\rho_{j,k} = w_k F_\rho + (1 - w_k)\rho_{j,k-1} \tag{89}$$

where F_ρ is an original event confidence updating function (e.g., sum and logistic). With Eq. 89, the data contribution of the participants who drop at the k^{th} iteration can still be kept (by $\rho_{j,k-1}$) to avoid the truth bias toward alive participants. w_k can be set as,

$$w_k = (\frac{|\mathcal{U}_{alive,k}|}{|\mathcal{U}|})^\alpha \tag{90}$$

where $\mathcal{U}_{alive,k}$ is the alive participants at the k^{th} iteration. When alive participants decrease with more iterations, w_k becomes smaller, reflecting that fewer participants should occupy lower weights. From our experiments, we find that $\alpha = 3$ is a proper setting.

4.2.2 Server-coordinated communication structure.

In Sec. 4.1, we assume that one participant u_i can establish a secure communication channel with every other participant u_k so as to transfer the secret share d_{ij}^k and s_{ij}^k. Hence, each participant needs to establish $n-1$ channels with others. Considering the sporadic property of the mobile connections, this may not be easy for a participant to keep so many channels stable in practice. To alleviate this issue, we convert the peer-to-peer communication structure to a server-coordinated one. In the server-coordinated structure, every participant first transmits all the data to the server, and then the server dispatches the desired data to the corresponding participant. In this way, each participant needs to establish *only one* secure channel to the server.

To ensure that the transmitted data will not be directly observed by the server, we leverage a public-key encryption system to encrypt the data before the transmission. In particular, each participant u_i first generates a pair of keys, the public key pk_i and the private key sk_i. The public key pk_i is sent to all the other participants (e.g., through the server) at the beginning of the crowdsensing campaign. For details, readers can refer to [1].

Then, for computing ρ_j, each participant u_i first transmits $E_i = \{\textit{Encrypt}(d_{ij}^k, pk_k)|k = 1 \cdots n\}$ to the server.[3] After receiving n participants' E_i, the server re-organizes the received data and sends $\hat{E}_k = \{\textit{Encrypt}(d_{ij}^k, pk_k)|i = 1 \cdots n\}$ to each participant u_k. Afterward, u_k decrypts the received data with her private key, obtains $\{d_{ij}^k|i = 1 \cdots n\}$, and then uploads $\hat{d}_j^k = \sum_i d_{ij}^k$ to the server. The server can then recover $d_j = \sum_i d_{ij}$ from participants' uploaded $\hat{d}_j^k$ and computes ρ_j accordingly.

4.2.3 (t, n)-Shamir secret sharing (SSS)

While the server-coordinated communication structure reduces the burden of secure channel establishing for mobile participants. It may still fail if a user loses the connection during the campaign and cannot link back. For example, suppose a participant u_i has sent E_i to the server and then quit the crowdsensing campaign (e.g., u_i's mobile device runs out of battery). Then, in Step 3, the server will not be able to receive $\hat{d}_j^i$ from u_i, and thus cannot recover d_j.

To address this pitfall, in practical deployment, we can leverage the threshold secret sharing method proposed by Shamir [21], namely (t, n)-*Shamir secret sharing (SSS)*. With (t, n)-SSS, the server only needs to receive

[3] If s_{ij} is needed (e.g., for the sum function), it can be encoded in E_i same as d_{ij}.

t ($t \leq n$) participants' $\hat{d}_j^i$ for recovering d_j. In particular, to leverage (t, n)-SSS to dispatch d_{ij}, we first create a $(t-1)$-polynomial:

$$D_{ij}(x) = d_{ij} + a_{ij1}x + a_{ij2}x^2 + \cdots + a_{ijt-1}x^{t-1} \tag{91}$$

where $a_{ij1}, \cdots, a_{ijt-1}$ are random numbers selected by u_i. Then, u_i dispatches $D_{ij}(k)$ to u_k. If we obtain more than t participants' $D_{ij}(k)$, according to linear algebra, we can infer d_{ij}.

Similar to Step 2 of our basic scheme, $\hat{d}_j^k = \sum_i D_{ij}(k)$ is uploaded to the server by each u_k. Then, in Step 3, after receiving more than t participants' $\hat{d}_j^k$, the server will be able to infer $d_j = \sum_i d_{ij}$ according to the *additive homomorphism property* of SSS [21].

Remark on our novelty. In the truth discovery part, the key advantage of our mechanism beyond literature is its robustness against connection loss. Preliminary privacy-preserving truth discovery papers rarely consider the connection loss issue [15]. Some work tries to deal with drop-out users by letting alive participants send extra information [38, 31]; however, when the connection condition is so bad that certain alive participants again lose connections during the extra information communication, this process would be uncontrolled and time-consuming. Recent work also adopts SSS to enhance connection robustness [34]. The basic idea is using the double-masking secure aggregation algorithms proposed by [1], and every participant needs *two* connections to do event confidence computation for one iteration. In comparison, our mechanism only needs every participant to connect *once* for one iteration of computation. Our numerical experiments (Sec. 6.1) will show that this connection reduction can lead to a significantly difference in the algorithm success probability (e.g., increasing the success probability from 1% to 99% under certain conditions).

4.3 Theoretical Analysis

4.3.1 Correctness

The process to calculate ρ_i in FedTruthFinder follows the original algorithm shown in Sec. 2. Hence, we can obtain the same aggregate truth results as the original algorithm, as long as the SSS scheme is valid. In this regard, the correctness of our algorithm is theoretically guaranteed.

4.3.2 Robustness to Connection Loss & Security

Setting t to a small value allows our mechanism to tolerate more users dropping the campaign due to connection losses. Meanwhile, a small t reduces the security level of our mechanism — if t participants collude with each other, they can recover the other participants' sensed data and locations, leading to privacy leakage.

Theorem 4.1. If there are $\leq n - t$ participants losing the connection in one iteration of ρ-computation, the server can learn the event confidence ρ_j.

Theorem 4.2. If $t'(< t)$ semi-honest users collude with each other, they cannot infer any other users' secret information.

The two theorems hold based on the property of (t, n)-SSS.

4.3.3 Complexity

We analyze the algorithm from both communication and computation complexity perspectives. Particularly, since the participant clients are more sensitive to the communication and computation overhead, our current analysis focuses on the client side, while the server part analysis is similar.

Communication Complexity - $O(nn_e)$. For each participant client, she needs to transfer n share pieces of the secret to the other participants and receive the corresponding shares from every other participant, so the complexity is $O(n)$ for one event. Suppose there are n_e events, the total communication complexity is $O(nn_e)$.

Computation Complexity - $O(nn_e)$. Each client needs to do two local computation processes. The first process is to generate the random coefficients for (t, n)-SSS and calculate the secret shares sent to all the other participants (Step 1), which is $O(nn_e)$. The second process is to do local summation (Step 2), which is also $O(nn_e)$. Hence, the total computation complexity is $O(nn_e)$.

5 Federated Trustworthiness Rank

While FedTruthFinder learns the integrated event truth in a privacy-preserving manner, it brings a challenge in justifying participants' trustworthiness. For example, to incentivize the crowdsensing participants, it is a common strategy to pay the high-trustworthy participants (i.e., high-quality sensing results) with higher incentives. However, in FedTruthFinder, the sensing quality of each participant, i.e., the trustworthiness score τ_i is kept at each participant side and unknown to the server. Hence, how to assess participants' trustworthiness is required and challenging for FedTruthFinder.

In this section, we first illustrate a concrete case to describe that τ_i cannot be directly known to the server, otherwise the server may infer which event u_i has sensed. As τ_i cannot be known to the server, we then design a secure ranking algorithm to let the server know every participant u_i's ranking position of τ_i among all the participants without leaking τ_i. Based on the ranked positions, the crowdsensing organizer can enable certain trustworthiness-aware incentive mechanisms, e.g., rewarding high-position participants with bonus, which can incentivize participants to compete with each other to get more high-quality sensed data [20].

5.1 Privacy Leakage by Trustworthiness τ_i

Here, we illustrate an example to show the risk of revealing τ_i to the server for leaking participant u_i's privacy.

Without the loss of generality, we assume that u_1's $\tau_1 = 0.9$, and other u_i's $\tau_i < 0.9$ $(i \neq 1)$. Suppose that one event e_j's $\rho_j = 0.9$ after truth discovery, then we can easily infer that u_1 has sensed the event e_j and the sensed result is 1. This reveals the fact that u_1 has visited the location of e_j, leaking u_1's location privacy.

Hence, participants cannot directly upload their τ_i to the server for incentive allocation. Next, we design a privacy-preserving method to enable trustworthiness-aware incentive allocation.

5.2 Secure Trustworthiness Leader-board

While revealing τ_i may leak participants' private information, we propose a secure ranking algorithm to learn a leader-board regarding participants' trustworthiness for facilitating trustworthiness-aware incentive allocation.

Secure ranking algorithms have been studied for decades; however, prior studies cannot be directly applied in our scenario for two reasons. First, the communication overheads are usually high. Second, prior studies mostly assume that all the network connections are stable for all the parties, but this is unrealistic for crowdsensing.

Our secure ranking algorithm generally follows the design of [22]. However, the original design [22] cannot tolerate any participants to lose the network connections. We thus enhance it to ensure that the ranking algorithm can still work when certain participants lose connections. The major steps of our federated trustworthiness leader-board generation mechanism are:

Step 1. First, we categorize all the participants into $(2t + 1)$ groups, and thus each group includes $n/(2t + 1)$ participants. We denote $gid(u)$ to refer to the group ID of participant u.

Step 2. For each user u_i, she shares τ_i, τ_i^2, ... , τ_i^{2t+1} with $(t + 1, 2t + 1)$-SSS to all the user groups. Specifically, a user u_j will receive the share piece regarding $gid(u_j)$, denoted as $\tau_{i_1}(gid(u_j))$, $\tau_{i_2}(gid(u_j))$, ... $\tau_{i_{2t+1}}(gid(u_j))$ for τ_i, τ_i^2, ... , τ_i^{2t+1}, respectively.

Step 3. For each user group g_k, it generates a random number $r_k(> 0)$ and shares r_k with $(t + 1, 2t + 1)$-SSS to all the user groups. That is, u_j will receive r_k's share regarding $gid(u_j)$, denoted as $r_k(gid(u_j))$.

Step 4. For each participant u_j, she calculates the following number with the $\tau_{i_k}(gid(u_j))$ received from u_i:

$$h_i(gid(u_j)) = \lambda(gid(u_j)) \sum_{k=1}^{2t+1} r_k(gid(u_j))\tau_{i_k}(gid(u_j)) \tag{92}$$

$$= \lambda(gid(u_j))\gamma(gid(u_j)) \tag{93}$$

where

$$\begin{pmatrix} 1 & 1 & 1^2 & ... & 1^{2t} \\ 1 & 2 & 2^2 & ... & 2^{2t} \\ ... & ... & ... & ... & ... \\ 1 & 2t+1 & (2t+1)^2 & ... & (2t+1)^{2t} \end{pmatrix}^{-1}$$

$$= \begin{pmatrix} \lambda(1) & \lambda(2) & \lambda(3) & ... & \lambda(2t+1) \\ ... & ... & ... & ... & ... \\ ... & ... & ... & ... & ... \\ ... & ... & ... & ... & ... \end{pmatrix}$$

Step 5. For each user group, we randomly select one participant u_j to share $\{h_i(gid(u_j))|i \in [1,n]\}$ with $(t+1,n)$-SSS to all the n participants. Each user u_k's received shares from all the groups are denoted as $\{h_i(g,k)|i \in [1,n], g \in [1,2t+1]\}$.

Step 6. For each participant u_k, she computes:

$$h'_i(k) = \sum_{g=1}^{2t+1} h_i(g,k), \quad \forall i \in [1,n] \tag{94}$$

Each u_k sends $\{h'_i(k)|i \in [1,n]\}$ to the server.

Step 7. After receiving at least $t+1$ participants' $\{h'_i(k)|i \in [1,n]\}$, the server can recover:

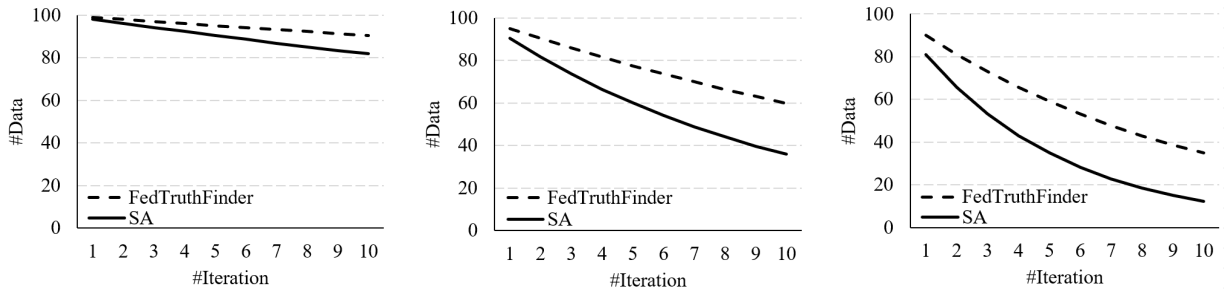
$$h_i = \sum_{k=1}^{2t+1} r_k \tau_i^k, \quad \forall i \in [1,n] \tag{95}$$

Step 8. The server ranks u_i according to h_i and the ranked list is the leader-board regarding trustworthiness τ_i.

Note that same as ρ-computation, we do not need to establish the peer-to-peer communication channels between every two participant clients and can use the crowdsensing server for coordination. To avoid redundancy, readers can refer to Sec. 4.2.2 for details.

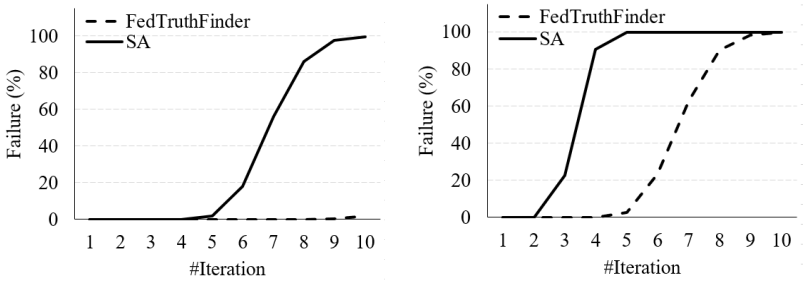
Remark on our novelty. The key improvement of our secure ranking algorithm compared to [22] is the enhanced robustness against participants' connection loss. In [22], every participant holds a r_i and we will randomly select $2t+1$ participants to share their r_i (Step 3) and h_i (Step 5). This process is easy to break if a selected online user (Step 3) loses the connection in Step 5. Our proposed algorithm first constructs user groups so that we only need at least one participant online in each group for both Step 3 and 5, reducing the failure possibility incurred by connection loss. It is worth noting that this algorithm can not only rank crowdsensing participants' trustworthiness, but also be applied to many other applications when privacy-preserving data ranking is needed under unstable network connections.

Remark on the ranked measurements. In the previous algorithm description, we suppose that τ_i needs to be ranked. In practice, crowdsensing organizers can use the same secure ranking mechanism to rank other key measurements of participants (e.g., the number of sensed events) to design better incentive mechanisms or participant recruitment strategies.



(a) $p_l = 0.01$ (b) $p_l = 0.05$ (c) $p_l = 0.1$

Figure 3: Number of data for each event's truth discovery by iterations.



(a) $p_l = 0.05$ (b) $p_l = 0.1$

Figure 4: Failure probability of truth discovery.

5.3 Theoretical Analysis

All the proofs are illustrated in Appendix.

5.3.1 Correctness

We first prove the correctness of our algorithm.

Lemma 5.1. $\sum_{k=1}^{2t+1} r_k(x)\tau_{i_k}(x)$ can be represented as:

$$h_i + a_{i1}x + a_{i2}x^2 + ... + a_{i2t}x^{2t}$$

where $h_i = \sum_{k=1}^{2t+1} r_k \tau_i^k$. [22]

Theorem 5.1. With $t+1$ participants' $h_i'(k)$, we can recover h_i.

Theorem 5.2. Ranking h_i is equivalent to ranking τ_i.

5.3.2 Robustness to Connection Loss

We analyze how our secure ranking algorithm can tolerate connection losses. We assume that before Step 2, there is no user connection loss.[4]

[4]If u_i loses the connection in Step 2 and cannot share τ_i^k with SSS, then there is no way to rank u_i's position because the server has no u_i's information.

Theorem 5.3. To finish Step 3-5, there needs at least one user online for each group. Suppose that every user has p_l probability to lose connection and there are n users, the success probability $\geq (1 - p_l^{\lfloor n/(2t+1) \rfloor})^{2t+1}$.

Theorem 5.4. To finish Step 6-8, $\geq t + 1$ users need to be online.

5.3.3 Security

Here, we analyze the security of our mechanism.

Theorem 5.5 If there are no more than t collusive participants, then these participants cannot recover all the other users' τ_i.

5.3.4 Complexity

We analyze the algorithm from communication and computation complexity perspectives for participant clients.

Communication Complexity - $O(tn)$. In Step 2, the communication overhead of one participant to send $\tau_i, \tau_i^2, ..., \tau_i^{2t+1}$ is $O(t^2)$, while each user received data is $O(tn)$. In Step 3, the complexity is $O(t)$. In Step 5, for sending data, the complexity is $O(n)$; for receiving data, the complexity is $O(tn)$. In Step 7, the complexity is $O(n)$. Combing them together, the communication complexity of the whole process is $O(tn)$ as $t < n$.

Computation Complexity - $O(tn)$. The main computation processes of each client include (1) calculating secret shares for $\tau_i, \tau_i^2, ..., \tau_i^{2t+1}$ with $(t + 1, 2t + 1)$-SSS in Step 2, which is $O(t^2)$, (2) calculating secret shares of r_k in Step 3, which is $O(t)$, (3) computing h_i in Step 4, which is $O(tn)$, and (4) calculating h_i' in Step 6, which is $O(tn)$. Hence, the final computation complexity is $O(tn)$.

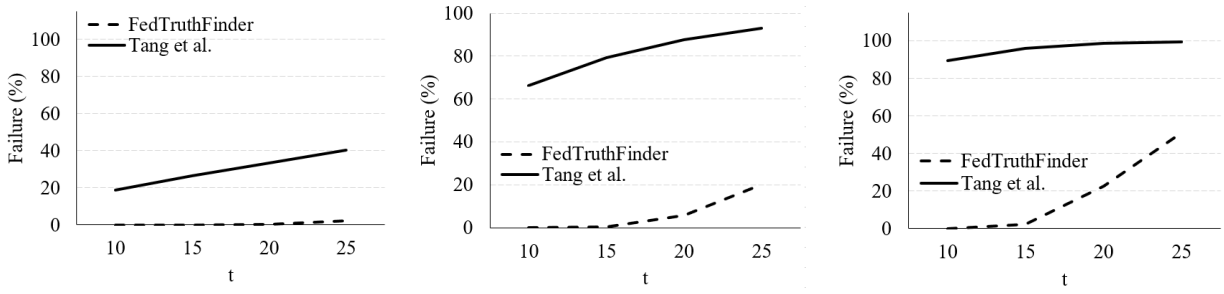
6 Evaluation

6.1 Numerical Analysis for Connection Loss

We have theoretically proven that our algorithm can learn the event confidence and trustworthiness ranking like the original centralized algorithms. This experiment then focuses on how the connection loss would impact FedTruthFinder quantitatively, since the unstable mobile network connection is a key characteristic for mobile crowdsensing. A practical mechanism should be able to fight against the unpredictable connection loss. In general, participants' connection loss may bring two types of negative impacts to the iterative truth discovery algorithm.

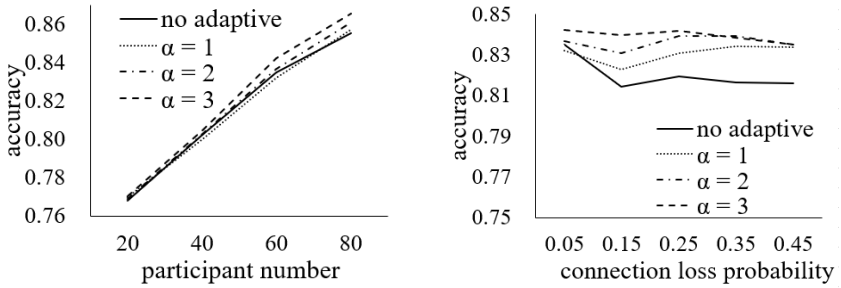
- **A small number of sensed data for truth discovery.** While FedTruthFinder can learn an aggregate truth as long as more than t participants are online, the data sources for the truth would be decreased. This would also affect the performance of the learned truth.
- **Possible failure of the whole algorithm.** If a large number of participants lose the connection and only fewer than t participants remain online, then the whole running process of FedTruthFinder would fail and no result can be learned.

Specifically, we conduct the numerical analysis for two parts of FedTruthFinder respectively, i.e., event confidence computation and participant trustworthiness ranking. We vary the probability of one participant losing the connection (denoted as p_l). If $p_l = 0.01$, a participant has 1% probability of dropping out of the crowdsensing campaign due to one-time connection loss. Then, if a participant needs to connect to the server for n times, it has $1 - (1 - p_l)^n$ probability to lose the connection. In the experiment, we test $p_l = 0.01/0.05/0.1$ to represent good/moderate/bad connection scenarios.



(a) $p_l = 0.01$ (b) $p_l = 0.05$ (c) $p_l = 0.1$

Figure 5: Failure probability of trustworthiness ranking.



(a) Participants number (b) Connection loss

Figure 6: Detection accuracy of FedTruthFinder.

6.1.1 Event Confidence Computation

To compare with FedTruthFinder, we consider the state-of-the-art way to do iterative truth discovery with an SSS-based secure aggregation (SA) protocol [1, 34], denoted as *SA*, which can also tolerate a certain level of participant connection loss. In brief, SA leverages a double-masking method to ensure that the truth discovery can run when some users lose the connection. However, not like FedTruthFinder which only needs a one-time connection for each participant to finish one iteration of ρ-computation, SA needs a two-time connection (double-masking).

Figure 3 shows the number of sensed data for truth discovery in each iteration for FedTruthFinder and SA (the total number of data is set to 100). Literature has shown that the number of iterations for truth discovery is often smaller than 10 [36] and thus we set the number of iterations up to 10. FedTruthFinder can always obtain more sensed data than SA as FedTruthFinder needs fewer connections. Especially, when the network connection condition is bad ($p_l = 0.1$), the performance improvement of FedTruthFinder over SA is more significant.

Figure 4 shows the algorithm failure probability (i.e., fewer than t users are online) for FedTruthFinder and SA (we set the number of participants to 100 and t to 50; we do not plot $p_l = 0.01$ as both methods are successful almost all the time). FedTruthFinder can significantly reduce the failure probability compared to SA. For example, when the network connection quality is moderate ($p_l = 0.05$), FedTruthFinder has around 99% probability to finish successfully for 10 iterations; however, SA has only around 1% probability. For the bad connection scenario ($p_l = 0.1$), SA will fail with more than 20% probability from iteration 3, while FedTruthFinder can keep working well until iteration 6. This reveals that, even if both FedTruthFinder and SA cannot finish all the ten iterations due to a bad network connection condition, FedTruthFinder can run a larger number of iterations, making the truth more reliable.

6.1.2 Participant Trustworthiness Rank

As none of the prior studies have addressed the privacy-preserving trustworthiness ranking problem, we cannot directly find a baseline method to compare. Meanwhile, our proposed trustworthiness ranking algorithm is inspired by the basic idea from [22] while significantly enhancing the capability to tolerate participants' connection loss. To this end, we compare FedTruthFinder and [22] when certain participants lose connections.

In federated trustworthiness ranking, t is the key parameter related to how many user groups are created, which significantly impacts the algorithm success probability (Theorem 5.3). Suppose the total number of users is 100, we set $t = 10/15/20/25$. The algorithm failure probability is shown in Figure 5. With the increase of t, the failure probability of FedTruthFinder rises. This fits our expectation as a larger t means that more user groups are generated and the user number per group is reduced. As FedTruthFinder needs at least one user online for each group, smaller user number per group means that the robustness against connection loss is weakened, leading to higher failure probability. Compared to [22], our algorithm significantly increases the success probability when connection is unstable. When $p_l = 0.1$ and $t = 10$, the failure probability of our ranking algorithm is 0.04%, but [22] is 96.18%. Hence, our algorithm could be an appropriate choice for ranking crowdsensing participants' trustworthiness scores considering the unstable mobile network connection environment.

6.2 Evaluation of Traffic Light Detection

We also test FedTruthFinder for traffic light detection, a representative crowdsensing task [16, 25]. We focus on the truth discovery accuracy and the runtime efficiency of FedTruthFinder, which has not been evaluated in the previous numerical analysis.

6.2.1 Data and Tasks

To evaluate FedTruthFinder on the traffic light detection task, we leverage a real-life open dataset including taxis' trajectories. Specifically, the dataset contains time-stamped GPS trajectories from 536 taxis in San Francisco, U.S. in one month of 2008 [18]. Following [16], we manually label 96 traffic light detection event positions using the Street View of Google Maps (Figure 7). Then, we randomly select some taxis as participants; their trajectories in the dataset are used to simulate their activities — if a taxi stops around an event's location, it may report the data. The report error rate (indicating trustworthiness) of each taxi is randomized in $[0, 0.5]$. The default participant number is 60 and the connection loss probability is 0.05. The event confidence function is set to 'logistic' as it performs better than 'sum'. t in SSS is set to half of the total participant number. To increase the randomness, each taxi randomly reports 20% of the events, and then each setting of the experiment is repeated by 50 times.

6.2.2 Experiment Platform

Our platform is an Alibaba cloud server with CPU of Intel Xeon Platinum 8163 (12 cores, 2.5GHz) and 24GB memory. The operating system is Ubuntu 20.04. FedTruthFinder is implemented by *Rust* 1.56. *Docker*[5] is adopted to simulate the crowdsensing server and participants.

6.2.3 Truth Discovery Accuracy

Figure 6(a) and 6(b) plot the accuracy regarding the number of participants and connection loss probability, respectively. Specifically, we compare FedTruthFinder with and without the adaptive truth updating technique (Sec. 4.2.1). For the adaptive updating, we try $\alpha = 1/2/3$ (Eq. 90), and find $\alpha = 3$ performs the best. The adaptive updating ($\alpha = 3$) can consistently improve the accuracy with different participant numbers and connection losses. Specifically, with more participants and higher connection losses, the improvement is more significant. When the

[5]https://www.docker.com/

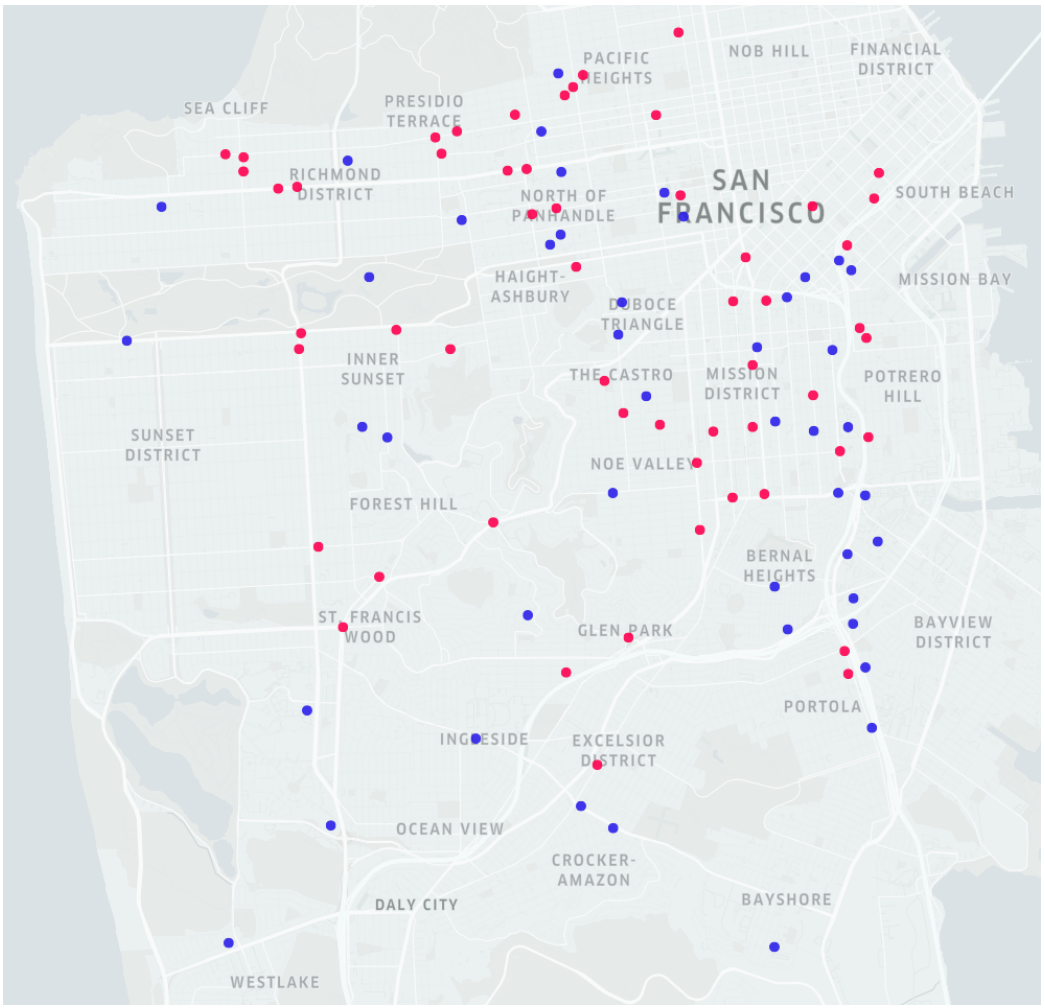


Figure 7: Traffic light event locations (red: true; blue: false). The red points represent the true event locations (i.e, with traffic lights) and the blue points mean the false event locations.

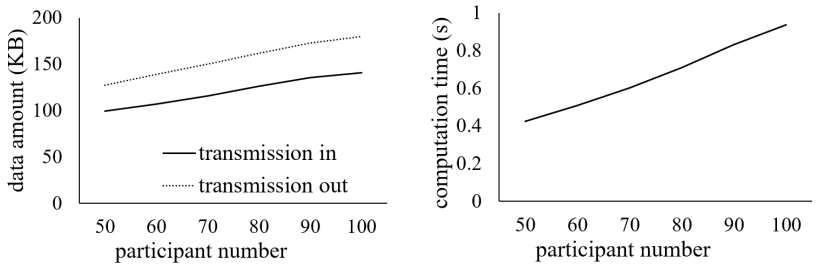
connection loss probability increases, the accuracy decreases gradually. This again verifies the effectiveness of FedTruthFinder over SA [34] — FedTruthFinder reduces the communication times per truth discover iteration compared to SA, which is conceptually equivalent to the reduction of connection losses in practice.

6.2.4 Runtime Efficiency

Figure 8(a) and 8(b) record each participant's data transmission amount and computation time, respectively. Note that the computation time is mostly spent in the truth finding step, while the trustworthiness ranking takes only $\sim$0.01s. The results show that the data transmission amount and computation time are both small, verifying the practicality of FedTruthFinder.

7 Related Work

Truth discovery is a traditional research direction as we may often receive diverse and even conflicting information about one event [7]. In the pioneering research [36], authors discuss the truth discovery problem when there



(a) Data transmission (b) Domputation time

Figure 8: Runtime efficiency of FedTruthFinder.

Table 15: Comparison of our work and representative related work. (NTP: No Third Party, AT: Assess Trustworthiness, CA: Collusion Attacks)

	Privacy Protection		**NTP**	**AT**	**Connection Loss**		**CA**
	Sensed Data	*Completed Tasks*			*Fault Tolerance*	*Bias Avoidance*	
[15]	√	×	×	×	×	×	×
[40]	√	×	×	×	√	×	×
[14]	√	×	√	×	√	×	×
[34]	√	×	√	×	√	×	√
[41]	√	×	√	×	√	×	√
[38]	√	×	×	×	√	×	×
Our Work	√	√	√	√	√	√	√

are many conflicting facts about one subject on different websites. Besides information from websites, truth discovery is also important in many other areas such as social sensing [26] and crowdsourcing [6, 33].

Mobile crowdsensing [39], as a particular type of crowdsourcing that needs workers to do location-based sensing tasks, would also face the truth discovery problem [24]. Meanwhile, privacy protection is also an important issue to consider in crowdsensing, especially for location privacy [4, 30, 27, 32, 29, 28]. Most prior research focuses on protecting crowdsensing participants' location privacy in task allocation [27, 32, 28] or for particular crowdsensing tasks such as missing data inference [30, 29].

Recently, some studies investigate the privacy-preserving truth discovery in crowdsensing [13, 15, 14, 38, 37, 34]. One research direction is applying data perturbation methods such as differential privacy to participants' sensed data [8, 9], but these methods degrade the truth finding accuracy. Another research direction follows the federated learning [35] paradigm that participants' raw data will not be directly sent to the server with certain encryption techniques, while the aggregation results (i.e., detected truths) can be accurately learned. However, the existing privacy-preserving truth discovery methods usually suffer from certain assumptions which may not stand in reality, e.g., online/non-concluding participants [13, 15, 14], or third-party non-concluding servers [38, 37]. Moreover, no prior work considers hiding participants' completed tasks or tracking participants' trustworthiness in a privacy-preserving manner, which has been addressed by our work.

Table 15 summarizes the characteristics of our work and representative related work published in top venues recently. In particular, our work is the **first** privacy-preserving crowdsensing truth discovery research that considers (i) *providing a feasible solution to participant trustworthiness assessment* when the trustworthiness scores are not revealed, and (ii) *hiding participants' completed tasks* to provide stronger privacy protection. Moreover, when dealing with the connection loss during the iterative crowdsensing truth discovery process, our work (i) proposes an adaptive event confidence updating function to reserve the data contributions of drop-out participants to avoid the truth bias toward alive participants, and (ii) designs an SSS-based scheme to defend

against participants' collusion attacks while ensuring the high communication efficiency.

8 Conclusion

In this paper, we propose *FedTruthFinder*, a crowdsensing federated truth discovery mechanism that can not only find aggregate truth from multiple participants' sensed data, but also rank participants' trustworthiness in a privacy-preserving manner. The primary characteristic of FedTruthFinder is its capability to tolerate network connection loss of participants in both event confidence calculation and participant trustworthiness ranking. As a byproduct, our proposed federated ranking algorithm can also serve other applications when the privacy-preserving data ranking is needed and the network connections are unstable. Following most related papers, this work assumes participants to be semi-honest; in the future, we would explore the more challenging scenario that participants may behave maliciously.

9 Appendix

9.1 Theoretical Proof

Proof of Lemma 5.1. It is clear that,

$$\sum_{k=1}^{2t+1} r_k(0)\tau_{i_k}(0) = \sum_{k=1}^{2t+1} r_k \tau_i^k \tag{96}$$

Besides, both $r_k(x)$ and $\tau_{i_k}(x)$ are t-degree polynomials, and thus the degree of $\sum_k r_k(x)\tau_{i_k}(x)$ is $2t$. □

Proof of Theorem 5.1. With Lemma 5.1, for N $(= 2t+1)$ groups, $\gamma(gid(u_j)) = \sum_{k=1}^{2t+1} r_k(gid(u_j))\tau_{i_k}(gid(u_j))$ (Step 4) is:

$$\begin{pmatrix} 1 & 1 & 1^2 & ... & 1^{2t} \\ 1 & 2 & 2^2 & ... & 2^{2t} \\ ... & ... & ... & ... & ... \\ 1 & N & N^2 & ... & N^{2t} \end{pmatrix} \begin{pmatrix} h_i \\ a_{i1} \\ ... \\ a_{i2t} \end{pmatrix} = \begin{pmatrix} \gamma(1) \\ \gamma(2) \\ ... \\ \gamma(N) \end{pmatrix}$$

then,

$$\begin{pmatrix} h_i \\ a_{i1} \\ ... \\ a_{i2t} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1^2 & ... & 1^{2t} \\ 1 & 2 & 2^2 & ... & 2^{2t} \\ ... & ... & ... & ... & ... \\ 1 & N & N^2 & ... & N^{2t} \end{pmatrix}^{-1} \begin{pmatrix} \gamma(1) \\ \gamma(2) \\ ... \\ \gamma(N) \end{pmatrix}$$

so,

$$h_i = \sum_{g=1}^{N} \lambda(g)\gamma(g)$$

In Step 5, $h_i(g) = \lambda(g)\gamma(g)$ is shared with $(t+1, n)$-SSS to all the participants from every group $g \in [1, 2t+1]$. Hence, according to the additive homomorphism property of SSS [21], we can easily recover h_i by receiving $t+1$ participants' $h_i'(k) = \sum_{g=1}^{2t+1} h_i(g, k)$. □

Proof of Theorem 5.2. As $\tau_i > 0$ and $r_k > 0$, $h_i = \sum_k r_k \tau_i^k$ will keep the same ranking as τ_i. □

Proof of Theorem 5.3. For Step 3 to 5, if there is at least one user in every group, then the process can continue. So the probability of failure incurred by one specific group g is all the users in g losing the connections,

i.e., $p^{n_g} \leq p_l^{\lfloor n/(2t+1) \rfloor}$ (n_g is the user number in g). So for g, the probability of at least one user online $\geq 1 - p_l^{\lfloor n/(2t+1) \rfloor}$. With $2t+1$ groups, the success probability $\geq (1 - p_l^{\lfloor n/(2t+1) \rfloor})^{2t+1}$. □

Proof of Theorem 5.4. This is based on the property of $(t+1, n)$-SSS in Step 5. □

Proof of Theorem 5.5. In Step 2, $\tau_i^k (k = 1...2t+1)$ is shared with $(t+1, 2t+1)$-SSS. So, if t participants collude, they can get at most $t \cdot (2t+1)$ equations when t participants are from t groups. However, the number of unknown parameters (including τ_i and t random coefficients for sharing each τ_i^k) is $t \cdot (2t+1) + 1$. Hence, these t collusive participants cannot recover other participants' τ_i. □

9.2 Mechanism Extension to Multi-class and Continuous-value Events

Multi-class Events. For a multi-class event (m classes), we can see it as m binary events, so that our method can be directly applied.

Continuous-value Events. For continuous-value events, following the literature, we may adopt other proper event confidence and participant trustworthiness updating functions such as CRH [34, 41]. Specifically, suppose that the discovered truth sensed value of a continuous event e_j is ρ_j, and u_i's sensed data of e_j is $\hat{\rho}_{ij}$, then the event truth (confidence) and participant trustworthiness updating functions can be:

$$\rho_j = \frac{\sum_{u_i \in \mathcal{U}_{e_j}} \tau_i \cdot \hat{\rho}_{ij}}{\sum_{u_i \in \mathcal{U}_{e_j}} \tau_i} \tag{97}$$

$$\tau_i = \log(\sum_{u_i \in \mathcal{U}} \sum_{e_j \in \mathcal{E}_{u_i}} \frac{(\rho_j - \hat{\rho}_{ij})^2}{|\mathcal{E}_{u_i}|}) - \log(\sum_{e_j \in \mathcal{E}_{u_i}} \frac{(\rho_j - \hat{\rho}_{ij})^2}{|\mathcal{E}_{u_i}|}) \tag{98}$$

where $\mathcal{U}_{e_j}$ is the set of users who sense e_j, and $\mathcal{E}_{u_i}$ is the set of events that u_i has sensed. For ρ-computation, following Sec. 4.1, we can just adapt d_{ij} and s_{ij} according to Eq. 97 (the participant $u_i \notin \mathcal{U}_{e_j}$ can still send $d_{ij} = s_{ij} = 0$ to protect her task completion information). For τ-computation, Eq. 98 requires $\sum_{u_i \in \mathcal{U}} \sum_{e_j \in \mathcal{E}_{u_i}} \frac{(\rho_j - \hat{\rho}_{ij})^2}{|\mathcal{E}_{u_i}|}$, which can be done with the same SSS-based method as ρ-computation. In particular, each participant u_i can send $\sum_{e_j \in \mathcal{E}_{u_i}} \frac{(\rho_j - \hat{\rho}_{ij})^2}{|\mathcal{E}_{u_i}|}$ by secret shares, and then the server can compute the sum in a privacy-preserving manner. In a word, for continuous-value events, our mechanism can still work without revealing each participant's raw sensed data and completed tasks.

References

[1] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth. Practical secure aggregation for privacy-preserving machine learning. CCS, 2017.

[2] D. Chai, L. Wang, K. Chen, and Q. Yang. Secure federated matrix factorization. IEEE Intelligent Systems, 2020.

[3] R. K. Ganti, F. Ye, and H. Lei. Mobile crowdsensing: current state and future challenges. IEEE Communications Magazine, 49(11):32–39, 2011.

[4] X. Han, L. Wang, and W. Fan. Is hidden safe? location protection against machine-learning prediction attacks in social networks. MIS Quarterly, 45(2):821–858, 2021.

[5] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon. Federated learning: Strategies for improving communication efficiency. arXiv:1610.05492, 2016.

[6] H. Li, B. Zhao, and A. Fuxman. The wisdom of minority: discovering and targeting the right group of workers for crowdsourcing. In WWW, pages 165–176, 2014.

[7] Y. Li, J. Gao, C. Meng, Q. Li, L. Su, B. Zhao, W. Fan, and J. Han. A survey on truth discovery. SIGKDD Explor. Newsl., 17(2):1–16, Feb. 2016.

[8] Y. Li, C. Miao, L. Su, J. Gao, Q. Li, B. Ding, Z. Qin, and K. Ren. An efficient two-layer mechanism for privacy-preserving truth discovery. KDD, 2018.

[9] Y. Li, H. Xiao, Z. Qin, C. Miao, L. Su, J. Gao, K. Ren, and B. Ding. Towards differentially private truth discovery for crowd sensing systems. ICDCS, pages 1156–1166, 2020.

[10] Y. Liu, Y. Kang, C. Xing, T. Chen, and Q. Yang. A secure federated transfer learning framework. IEEE Intelligent Systems, 35(4):70–82, 2020.

[11] H. Ma, D. Zhao, and P. Yuan. Opportunities in mobile crowd sensing. IEEE Communications Magazine, 52(8):29–35, 2014.

[12] C. Meng, W. Jiang, Y. Li, J. Gao, L. Su, H. Ding, and Y. Cheng. Truth discovery on crowd sensing of correlated entities. In Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems, pages 169–182, 2015.

[13] C. Miao, W. Jiang, L. Su, Y. Li, S. Guo, Z. Qin, H. Xiao, J. Gao, and K. Ren. Cloud-enabled privacy-preserving truth discovery in crowd sensing systems. In SenSys, 2015.

[14] C. Miao, W. Jiang, L. Su, Y. Li, S. Guo, Z. Qin, H. Xiao, J. Gao, and K. Ren. Privacy-preserving truth discovery in crowd sensing systems. ACM Transactions on Sensor Networks, 15:1 – 32, 2019.

[15] C. Miao, L. Su, W. Jiang, Y. Li, and M. Tian. A lightweight privacy-preserving truth discovery framework for mobile crowd sensing systems. INFOCOM, pages 1–9, 2017.

[16] R. W. Ouyang, M. Srivastava, A. Toniolo, and T. J. Norman. Truth discovery in crowdsourced detection of spatial events. IEEE transactions on knowledge and data engineering, 28(4):1047–1060, 2015.

[17] D. Peng, F. Wu, and G. Chen. Data quality guided incentive mechanism design for crowdsensing. IEEE Transactions on Mobile Computing, 17:307–319, 2018.

[18] M. Piorkowski, N. Sarafijanovic-Djukic, and M. Grossglauser. CRAWDAD dataset epfl/mobility (v. 2009-02-24). Downloaded from `https://crawdad.org/epfl/mobility/20090224`, Feb. 2009.

[19] V. Primault, A. Boutet, S. B. Mokhtar, and L. Brunie. The long road to computational location privacy: A survey. IEEE Communications Surveys & Tutorials, 21:2772–2793, 2019.

[20] S. Reddy, D. Estrin, M. Hansen, and M. Srivastava. Examining micro-payments for participatory sensing data collections. UbiComp, 2010.

[21] A. Shamir. How to share a secret. Communications of the ACM, 22(11):612–613, 1979.

[22] C. Tang, G. Shi, and Z. Yao. Secure multi-party computation protocol for sequencing problem. SCIENTIA SINICA Informationis, 41(7):789–797, 2011.

[23] X. Tang, C. Wang, X. Yuan, and Q. Wang. Non-interactive privacy-preserving truth discovery in crowd sensing applications. INFOCOM, pages 1988–1996, 2018.

[24] D. Wang, T. Abdelzaher, and L. M. Kaplan. Surrogate mobile sensing. IEEE Communications Magazine, 52:36–41, 2014.

[25] D. Wang, L. Kaplan, T. Abdelzaher, and C. C. Aggarwal. On credibility estimation tradeoffs in assured social sensing. IEEE Journal on Selected Areas in Communications, 31(6):1026–1037, 2013.

[26] D. Wang, L. Kaplan, H. Le, and T. Abdelzaher. On truth discovery in social sensing: A maximum likelihood estimation approach. In IPSN, pages 233–244, 2012.

[27] L. Wang, D. Yang, X. Han, T. Wang, D. Zhang, and X. Ma. Location privacy-preserving task allocation for mobile crowdsensing with differential geo-obfuscation. WWW, 2017.

[28] L. Wang, D. Yang, X. Han, D. Zhang, and X. Ma. Mobile crowdsourcing task allocation with differential-and-distortion geo-obfuscation. IEEE Transactions on Dependable and Secure Computing, 18:967–981, 2021.

[29] L. Wang, D. Zhang, D. Yang, B. Y. Lim, X. Han, and X. Ma. Sparse mobile crowdsensing with differential and distortion location privacy. IEEE Transactions on Information Forensics and Security, 15:2735–2749, 2020.

[30] L. Wang, D. Zhang, D. Yang, B. Y. Lim, and X. Ma. Differential location privacy for sparse mobile crowdsensing. In ICDM, pages 1257–1262. IEEE, 2016.

[31] T. Wang, C. Lv, C. Wang, F. Chen, and Y. Luo. A secure truth discovery for data aggregation in mobile crowd sensing. Secur. Commun. Networks, 2021:2296386:1–2296386:15, 2021.

[32] Z. Wang, J. Hu, R. Lv, J. Wei, Q. Wang, D. Yang, and H. Qi. Personalized privacy-preserving task allocation for mobile crowdsensing. IEEE Transactions on Mobile Computing, 18:1330–1341, 2019.

[33] J. Whitehill, P. Ruvolo, T. Wu, J. Bergsma, and J. Movellan. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In NeurIPS, 2009.

[34] G. Xu, H. Li, S. Liu, M. Wen, and R. Lu. Efficient and privacy-preserving truth discovery in mobile crowd sensing systems. IEEE Transactions on Vehicular Technology, 68:3854–3865, 2019.

[35] Q. Yang, Y. Liu, T. Chen, and Y. Tong. Federated machine learning: Concept and applications. ACM Transactions on Intelligent Systems and Technology, 10(2):12, 2019.

[36] X. Yin, J. Han, and S. Y. Philip. Truth discovery with multiple conflicting information providers on the web. IEEE Transactions on Knowledge and Data Engineering, 20(6):796–808, 2008.

[37] C. Zhang, C. Xu, L. Zhu, Y. Li, C. Zhang, and H. Wu. An efficient and privacy-preserving truth discovery scheme in crowdsensing applications. Computers & Security, 97:101848, 2020.

[38] C. Zhang, L. Zhu, C. Xu, X. Liu, and K. Sharif. Reliable and privacy-preserving truth discovery for mobile crowdsensing systems. IEEE Transactions on Dependable and Secure Computing, 18:1245–1260, 2021.

[39] D. Zhang, L. Wang, H. Xiong, and B. Guo. 4w1h in mobile crowd sensing. IEEE Communications Magazine, 52(8):42–48, 2014.

[40] Y. Zheng, H. Duan, and C. Wang. Learning the truth privately and confidently: Encrypted confidence-aware truth discovery in mobile crowdsensing. IEEE Transactions on Information Forensics and Security, 13:2475–2489, 2018.

[41] Y. Zheng, H. Duan, X. Yuan, and C. Wang. Privacy-aware and efficient mobile crowdsensing with truth discovery. IEEE Transactions on Dependable and Secure Computing, 17:121–133, 2020.

Federated Ensemble Learning: Increasing the Capacity of Label Private Recommendation Systems

Meisam Hejazinia, Dzmitry Huba, Ilias Leontiadis, Kiwan Maeng, Mani Malek,
Luca Melis, Ilya Mironov, Milad Nasr, Kaikai Wang, Carole-Jean Wu
Meta Platforms, Inc.*

Abstract

Despite proven effectiveness of federated learning (FL) as a solution to private model training, FL has had limited success in the domains of ranking and recommendation systems. This is primarily due to the fact that modern recommendation systems, particularly in the context of on-line advertising, rely on large neural networks that cannot be feasibly trained on user devices. However, given increasing privacy regulations and evolving users' preferences, it is imperative for advertising platforms to invest in private learning solutions like FL that support training highly accurate models with strong privacy guarantees.

In this paper, we propose Federated Ensemble Learning (FEL) as a solution to address the large memory requirement for recommendation systems subject to label privacy. FEL enables scalable label-private recommendation model training by simultaneously training multiple smaller FL models on disjoint carefully selected clusters of client devices. The output of these models is aggregated with a neural network trained either on server-side data or via a second stage of private on-device training. Our experimental results demonstrate that FEL leads to 0.43–2.31% model quality improvement over traditional on-device federated learning — a significant improvement for ranking and recommendation system use cases.

1 Introduction

Federated learning (FL) has emerged as an effective approach to address consumer privacy needs by allowing edge devices to collaboratively train a machine learning (ML) model, while the raw data samples remain on-device [5, 20]. While FL has been deployed for a variety of machine learning tasks, such as smart keyboard [1], personalized assistant services [17], computer vision [27], healthcare [45], it has seen limited adoption for ranking and recommendation tasks [32, 42]. This is due to the fact that constrained client resources in FL prevent training a large, high-accuracy recommendation model (often in the order of GBs or TBs [30, 53]), while meeting the privacy requirement. However, recommendation models need to achieve a strong user privacy and a high accuracy at the same time [1],

Furthermore, in contrast to conventional privacy settings of FL, a unique characteristic of modern recommendation systems is that models are usually trained on public features but with private labels. For instance, features

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

*Authors are listed alphabetically.

[1] A recent study from Baidu [53] indicates that even a 0.1% accuracy drop is considered unacceptable for its ranking and recommendation tasks.

such as user profiles and product catalogs are known to advertising platforms, but the labels, i.e., transaction details, are considered third-party (private) data. Matching cross-site or cross-app data is privacy-sensitive and can be subject to regulation and user-device policies. This new label-only privacy setting has gained significant interest from both academia and industry recently [13, 33].

Focusing on the new label-only privacy requirement, we propose a new FL framework called Federated Ensemble Learning (FEL). FEL addresses the key limitations of FL for ranking systems in the novel setting of label privacy.

FEL proceeds in three stages: first, users are clustered into groups of similar behaviors using public features; second, a leaf model is learned for every cluster of users, simultaneously with other clusters using FL with global differential privacy (DP); and third, the leaf models (or parts of them) are concatenated to form a back-bone for a larger server model that can be trained on public data on the server or on private data on user devices. The insight behind FEL is that if client device clusters are appropriately formed, leaf models can learn distinct, potentially complementary, representations from each cluster, which are later combined to enhance the overall prediction capability. At each stage, the FEL framework ensures that the resulting ensemble satisfies the privacy goals by adapting the noise characteristics of DP for each leaf. We use composition theory to estimate the privacy cost of the ensemble aggregation layer and to automatically tune the noise added.

FEL excels when the number of observations per user is small, but the number of users is high (e.g., in the order of billions) — a setting that is common for recommendation and ranking tasks. We have deployed FEL in the production environment of a large-scale ranking system. We show that the deployed recommendation task achieves 0.43% precision gain compared to the vanilla FL baseline in the production environment, which is significant in the context of production ranking and recommendation systems. We observe significant gains of 1.55–2.31% using the open-source datasets, Ads click prediction [46] and image classification [29]. In addition, we show that FEL outperforms standard FL in the presence of differential privacy (DP) noise by 0.66–1.93%.

2 Background: Federated Learning and Privacy Assumptions

FL trains a model collaboratively using client devices without requiring the raw data to be shared with service providers. However, the key constraint in using vanilla FL training for recommendation and ranking tasks is the limited model size it can support. In many federated recommendation and ranking tasks, the input space is large, e.g., over 1,000 features, and the data distribution is multi-modal. To achieve high accuracy in this setting, we have to increase the overall model capacity. However, FL can be applied only to sufficiently small models that can be transmitted and trained on end-user devices.

Prior work studied increasing model capacity by leveraging client heterogeneity: training larger models on devices that are more powerful, while sending smaller models to less capable devices. The smaller model can be a subsampled model that is later aggregated to the supernet [6, 19, 9], or a different model that later transfers its learned knowledge to the larger model with knowledge distillation [26, 23]. These approaches still limit the model capacity as the model size is capped by the most powerful client devices that still cannot train GB-size models.

Privacy Assumptions of FEL: FEL targets the label-only privacy setting, where the input is public (i.e., accessible to the service provider) while the label is private. Several advertising, recommendation, and survey/analytics applications fall into this category [13, 33, 37, 43].

Many recommendation tasks use features that are public from the perspective of the recommender system. Public features include user attributes that are explicitly shared at sign-up time (e.g., age or gender) [46], externally observable user behavior (e.g., public movie reviews) [15], or item information that is provided by the item vendors [38]. On the other hand, the labels of recommendation tasks may be considered private, e.g., user conversion behavior [40, 13]. It is also possible to use a mixture of public and private features. FEL can be further extended to incorporate private features (Section 3).

While user labels must be kept private, i.e., unknown to the service provider, there can be opt-in users who consent to sharing their private label information with the service provider to improve the service quality. FEL does not require the presence of opt-in users; however, having some opt-in user population can simplify the training algorithm (Section 3) and lower the privacy cost (Section 3.2).

To avoid statistical inference attacks targeting FL, differentially private (DP) noise is added either on device or during the aggregation step [12, 48, 51]. We target the user-level differential privacy, in which the trained model weights are similarly distributed with or without a particular user [34]. We assume an honest-but-curious provider for training FL, which uses either hardware-based encryption in a trusted enclave, or software-based encryption via multiparty computation for FL aggregation [36, 25].

3 Proposed Design: Federated Ensemble Learning (FEL)

Rather than training one model across all users, FEL proposes to train a distinct leaf model per user cluster and later aggregate the leaf models on the server to obtain a larger-capacity model. Ensemble methods similar in spirit have shown promising potential with classical machine learning algorithms, e.g., in the form of AdaBoost, random forest, and XGBoost [22]. We explored different variations in how the clients are clustered and how the leaf models are aggregated.

3.1 Federated Ensemble Learning

Figure 1 illustrates the proposed design of Federated Ensemble Learning (FEL). First, we cluster the clients into a desired number of clusters (Figure 1, Step 1). Then, we train a leaf model for each cluster using traditional FL (Figure 1, Step 2). After training, the server uses the ensemble of the leaf models for inference requests (Figure 1, Step 3). On each inference request, the server passes the public input features through all the leaf models. Then, the output of each leaf model and/or the output of the last hidden layer of each leaf model is passed to the ensemble aggregation layer, which outputs the final prediction. Below, we lay out how FEL forms clusters and implements ensemble aggregation, and how FEL can be extended to incorporate private features.

Forming Clusters Client clusters can be formed based on distinctive characteristics of the users, e.g., a user's age, location, or past preferences. Clusters can also be obtained by simple hashing, or through popular clustering approaches such as k-means or Gaussian mixture methods. Marketers have been forming clusters of clients to target each cluster more effectively, and those well-studied clustering methods can be adopted [4].

Rudimentary Ensemble Aggregation: A simplest way to implement ensemble aggregation is to collect the prediction output of each leaf model and perform typical aggregation methods, such as *mean*, *median*, or *max*, to generate the final prediction. This approach is similar to bagging technique leveraged in random forest [44].

Neural Network (NN)-based Ensemble Aggregation: NN-based ensemble aggregation uses a separate neural network that takes in the prediction and the output of the last hidden layer of all leaf models as input to generate the final prediction. We call this additional neural network model *the over-arch model*. The over-arch model is trained after all the leaf models are trained. The over-arch model can be trained in several different ways. In the presence of opt-in users, the server can simply use them to train the over-arch model. Otherwise, the server can again use FL to train the over-arch model on each client, where the server sends the output of the leaf models to the client, and the client uses it as an input to train the over-arch model locally.

Extending to Private Features FEL can be extended to support private features only known to the clients. This can be done by (privately) training a separate leaf model (private leaf model) that only takes in private client-side features. The output of the private leaf model is used as an input to the ensemble aggregation layer along with other leaf models. When the private leaf model is used, part of the inference must happen on-device instead of entirely on the server. Specifically, the server sends outputs of the leaf models to each client, which ensembles them with its private leaf model output on-device for prediction. By performing the ensemble on-device, the input/output of the private leaf model is never exposed to the server. The private leaf model is trained with conventional FL as well.

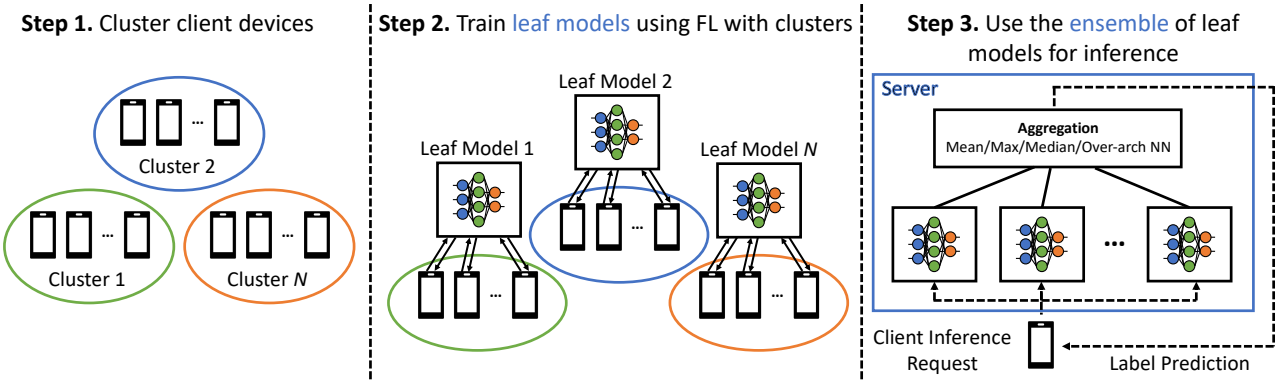


Figure 1: Federated Ensemble Learning architectures

3.2 Privacy Analysis for Federated Ensemble Learning

Ensuring data privacy is the utmost requirement and this work is the first to conduct privacy analysis for federated ensemble learning. Differential privacy for federated learning bounds how much the distribution of model parameters change between two datasets that differ in labels contributed by a single user [34]. We use a generalization of differential privacy [11] based on the Rényi divergence:

Definition 1 (Renyi Differential Privacy (RDP) [35]). A *randomized mechanism* M *with domain* D *is* (α, ϵ)*-RDP with order* $\alpha \in (1, \infty)$ *iff for any two neighboring datasets* $D, D' \in \mathrm{D}$*:*

$$\mathrm{D}_\alpha(\mathrm{M}(D)||\mathrm{M}(D')) := \frac{1}{\alpha - 1} \log E_{x \sim \mathrm{M}(D')} \left[\left(\frac{\Pr[\mathrm{M}(D) = x]}{\Pr[\mathrm{M}(D') = x]} \right)^\alpha \right] \leq \epsilon.$$

FEL is a multi-step process. To analyze the privacy bounds of a multi-step process, a common approach in differential privacy is to evaluate each process individually, then calculate the overall privacy bounds by composing all of the steps. In particular, we focus on two main composition theorems in differential privacy, sequential and parallel composition [10]. Formally we define:

Theorem 1 (Sequential Composition). *Let there be n RDP-mechanisms* M_i *with* (α, ϵ_i)*-RDP when being computed on a dataset* D *of the input domain* D*. Then, the composition of n mechanisms* $\mathrm{M}(\mathrm{M}_1(D), \ldots, \mathrm{M}_n(D))$ *is* $(\alpha, \sum_{i=1}^n \epsilon_i)$*-RDP*

Theorem 2 (Parallel Composition). *Let there be n RDP-mechanisms* M_i *with* (α, ϵ_i)*-RDP when being computed on disjoint subset* D_i *of the input domain* D*. Then, the composition of n mechanisms* $\mathrm{M}(\mathrm{M}_1(D), \ldots, \mathrm{M}_n(D))$ *is* $(\alpha, \max_{i=1}^n \epsilon_i)$*-RDP*

Sequential composition considers the case where a task uses the same users (even if different steps use different parts of the user's data) in different steps of an algorithm. For example, if the algorithm has four steps each with a privacy cost ε and uses the same users in all the steps, the total privacy cost of the algorithm would become 4ε. Parallel composition considers a case where each step is applied to different users. From the earlier example, if four disjoint sets of users were used for the four steps, the overall privacy cost would be $\max(\varepsilon_1, \varepsilon_2, \varepsilon_3, \varepsilon_4)$, where ε_k is the privacy cost of the k-th step. The compositions are tracked using RDP, which leads to tighter bounds than (ϵ, δ)-DP.

When analyzing the privacy cost of training the leaf models, the parallel composition theorem applies as each leaf model is trained with a disjoint set of users. If training a leaf model with cluster $i \in \{1, \ldots, N\}$ has a privacy cost e_i, the privacy cost of the entire leaf model is $e_{\text{leafs}} = \max(e_1, \ldots, e_N)$.

The privacy cost of the ensemble aggregation layer e_{agg} depends on the aggregation method that is used (mean, max, median, or NN-based). When using the mean, max, and median ensemble aggregation, $e_{\text{agg}} = 0$ as both theorems show that privacy cost increases only when the additional step (ensemble aggregation) uses user data. In this case, the total privacy cost e_{tot} is simply e_{leafs}.

When the NN-based approach is used, however, user data is used to train the over-arch NN layer and $e_{\text{agg}} > 0$. Depending on exactly how the over-arch NN layer is trained, e_{tot} can be calculated in the following way. First, if the over-arch NN layer is trained with the users that trained the leaf models, sequential composition theorem applies ($e_{\text{tot}} = e_{\text{agg}} + e_{\text{leafs}}$). Second, if the over-arch NN layer is trained with a completely different set of users, the parallel composition theorem applies ($e_{\text{tot}} = \max(e_{\text{agg}}, e_{\text{leafs}})$). Finally, if the over-arch NN layer is trained with opt-in users, $e_{\text{agg}} = 0$ because no private data is used, and $e_{\text{tot}} = e_{\text{leafs}}$. In all cases, the privacy cost of FEL does not significantly deteriorate over the vanilla FL (which is similar to e_{leafs}).

4 Evaluation

In this section we would like to provide more details about our experiments. We first provide details about each dataset that was used. We then provide details about the model architectures that we used for each dataset.

4.1 Datasets for FEL

We explored three datasets for FEL: An internal production dataset that represents a real-world ads-ranking system, an external dataset that addresses a similar ads-ranking task, and an image classification dataset.

4.1.1 Production Dataset

Production dataset is an internal dataset that captures whether a user installs a mobile application after being shown a relevant advertisement item. A few hundred features are used as an input (the exact number cannot be disclosed) to predict a binary label (install/not-install). All the input features are public. For training, we use advertisement data from a random sample of 35 million users over a period of one month. Randomly selected 15 million users from the following week were used for testing.

4.1.2 Taobao CTR Dataset

The Taobao dataset contains 26 million interactions (click/non-click when an Ad was shown) between 1.14 million users and 847 thousand items across an 8-day period. The dataset uses 9 user features (e.g., gender or occupation), 6 item features (e.g., price or brand), and two contextual features (e.g., the day of week), which we assume to be all public to the service provider.

In the Taobao CTR dataset, 16 out of the 17 features are sparse, with a categorical value encoding instead of a continuous, floating point value. While server-based recommendation models use large embedding tables to convert these sparse features into a floating point embedding [54, 38, 8], training such embedding tables on device is complicated because of the large memory capacity requirement (e.g., in the order of GB to TB [53, 2, 52, 30]) and can leak private information more easily through gradients [40]. Thus, we assume an architecture where embedding tables are pre-trained with opt-in users and are hosted on the server, while the rest of the model is trained with FEL using sparse features translated through the pre-trained tables. We randomly selected 10% of the users as opt-in.

Our setup cannot achieve the accuracy that can be reached when we fully train the embedding tables, as we pre-train the embeddings and fix their weight during FL. However, our setup represents a practical FL setup where training embedding tables on-device is prohibitive, due to client resource limitations [39] and privacy concerns [40].

4.1.3 CelebA Smile Prediction Dataset

While FEL is originally designed for recommendation and ranking tasks, we study its generality to non-recommendation models with CelebFaces Attributes Dataset (CelebA) [29]. CelebA consists of $200,288$ images

belonging to $9,343$ unique celebrities. Each image has 40 binary facial attribute annotations (e.g., bald, long hair, attractive, etc) and covers large pose variations and backgrounds. We defined distinguishing between smiling/non-smiling images as our target task.

The Taobao dataset contains 26 million interactions (click/non-click when an Ad was shown) between 1.14 million users and 847 thousand items across an 8-day period. The dataset uses 9 user features (e.g., gender or occupation), 6 item features (e.g., price or brand), and two contextual features (e.g., the day of week), which we assume to be all public to the service provider. The details on how we preprocess the dataset can be found in the appendix.

4.2 Model Architectures

Production/Taobao Dataset: For recommendation datasets (production/Taobao CTR), we use a model that consists of 3 fully-connected hidden layers. The number of units at each hidden layer is decreasing exponentially with a parameter K. For instance, if $K = 4$ and the input layer has 512 features, our neural network would have $[512, 128, 32, 8, 1]$ neurons. For each dataset, we tune K to obtain a resulting model of approximately 10MB. By doing so, it allows us to train a neural network even on older, low-tier devices with more limited memory capacity. ReLu is used as an activation function after each layer apart from the last one, where Sigmoid and binary cross-entropy was used.

For both datasets, we use synchronous FL with FedAvg [34]. We used the following hyperparameters for the Taobao dataset from an extensive hyperparameter search: client batch size of 32, 5 local epochs, 4096 clients per round, and a learning rate of 0.579 with SGD. Clients are selected at random and each only participates once (1 global epoch). The production dataset used similar hyperparameters.

For Taobao dataset's server-side pre-trained embedding table, we use an embedding dimension of 32, and train it with the 10% opt-in users for 1 epoch using AdaGrad optimizer with learning rate of 0.01.

CelebA Dataset: For CelebA, we follow the setup of prior work [39] and use a four layer CNN with dropout rate of 0.1, stride of 1, and padding of 2. We preprocess all images in train/validation/test sets; each image is resized and cropped to 32×32 pixels, then normalized by 0.5 mean and 0.5 standard deviation. We use asynchronous FL with a client batch size of 32 samples, 1 local epoch, 30 global epochs, and a learning rate of 0.899 with SGD.

4.3 Evaluation Methodology

Our evaluation aims to answer the following questions:

- Can FEL improve the model prediction quality over vanilla FL? [Section 4.4]
- How do different ensemble aggregation methods affect the model accuracy? [Section 4.4]
- How do different clustering methods affect the model accuracy? [Section 4.5]
- How does FEL affect privacy compared to vanilla FL? [Section 4.6]

To answer these questions we used the three datasets presented in Section 4.1. To study recommendation and ranking tasks, we used a production dataset and an open-source, Taobao's Click-Through-Rate (CTR) prediction dataset [24]. To study the effect of FEL on non-recommendation use-cases, we additionally studied the LEAF CelebA Smile Prediction dataset [29]. More details about these datasets and their associated model architecture can be found in Section 4.1.

Both the FL baseline and the FEL leaf models used the same set of hyperparameters. The FL baseline is trained using all the available client data. In FEL, the client data is clustered, and one leaf model is trained for each cluster. We vary the number of clusters from 3–10 and evaluate different clustering methods. When training the over-arch NN layer, a small subset of opt-in users is used.

Table 16: Explanation of different cluster methods in Figure 2 (right).

Dataset	Config	Feature	# clusters
Production	Clustering 1	Age	5
	Clustering 2	App	5
	Clustering 3	Location	4
	Clustering 4	Click ratio	10
Taobao [46]	Clustering 1	Age	7
	Clustering 2	Consumption	4
	Clustering 3	City level	5
CelebA [29]	Clustering 1	# Attributes	3
	Clustering 2	K-means	3
	Clustering 3	K-means	5

Table 17: FEL's prediction accuracy improvement over the baseline FL for different datasets. Following common practice of each dataset, Taobao uses AUC and CelebA uses accuracy as their metric. Production data's baseline accuracy is not disclosed.

	Production	Taobao [46] AUC	CelebA [29] accuracy
Baseline FL	-	0.5418 [3]	90.75
FEL (Mean Best)	(+0.27%)	0.5522 (+1.92%)	91.68 (+1.02%)
FEL (Median Best)	(+0.29%)	0.5459 (+0.74%)	91.35 (+0.66%)
FEL (Max Best)	(-0.06%)	0.5418 (-0.1%)	91.46 (+0.78%)
FEL (NN-based Best)	**(+0.43%)**	**0.5544 (+2.31%)**	**92.16 (+1.55%)**

4.4 Prediction Quality Improvement of FEL

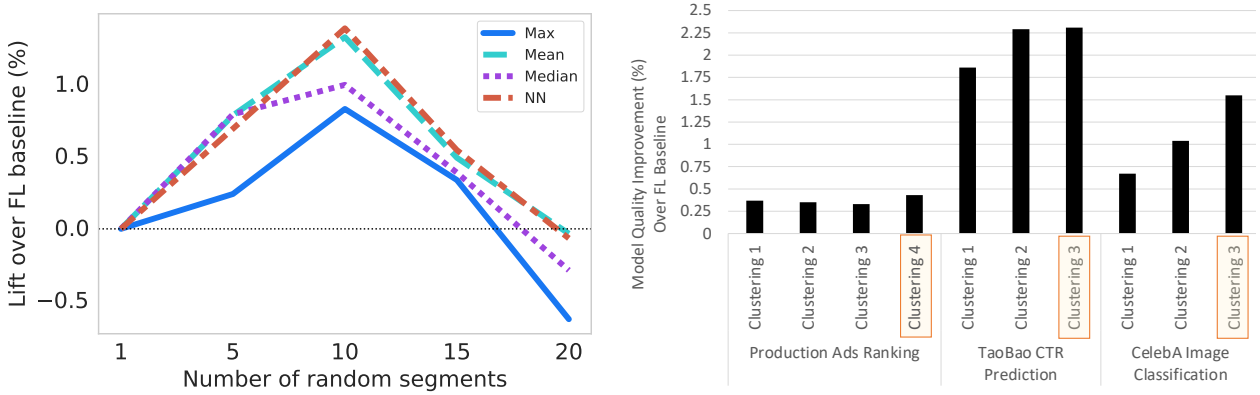
Overall, FEL achieves **0.43%** and **2.31%** prediction quality improvement over vanilla FL for production and Taobao datasets, respectively – a significant improvement for ranking and recommendation system use cases[2]. For non-recommendation tasks (CelebA), FEL shows similar improvement of **1.55%**, indicating that FEL can be generalized to non-recommendation use-cases as well. Table 17 summarizes the resulting prediction quality improvement of FEL compared to the baseline FL. We used accuracy for CelebA [39] and ROC-AUC (AUC) for Taobao [46]. We used normalized entropy for the production dataset, which we cannot disclose and only show the relative improvement. For different ensemble aggregation methods, we vary the clustering methods and report the best-accuracy results. Among the different ensemble aggregation methods, adding an over-arch NN layer provided the best prediction quality improvement, followed by mean and median.

4.5 Prediction Quality Improvement of Different Clustering Methods

Effects of the Number of Clusters: To understand the effect of the number of client clusters in the final model quality improvement, we varied the number of clusters in the Taobao dataset while using random clustering. Figure 2 (left) summarizes the result. There is an optimal setting for the number of clusters used in FEL. Going beyond the optimal setting for the number of clusters results in worse model accuracy. When the number of clusters is too small, the final model capacity is limited as there are not enough leaf models to ensemble. If the number of clusters is too large, each leaf model cannot learn enough information as the clients in each cluster are too few. The optimal number of clusters depends on the number of available devices that participate within each

[2] [53] mentioned 0.1% model quality improvement as significant and [50] considered 0.23% as impactful in similar recommendation and ranking use-cases.

[3] Taobao's baseline AUC is 0.26% less than the baseline FL result presented at [40], potentially due to simpler model architecture and freezed pre-trained embedding tables.



(a) (b)

Figure 2: Accuracy improvement for different number of clusters (segments) for each ensemble aggregation method (left), and different clustering methods when using over-arch NN layer (right). Different clustering methods are explain in Table 16

Table 18: Taobao dataset with DP. Percentage of FEL's accuracy improvement over the FL baseline with the same level of DP noise is shown. Table 16 explains the clustering configurations.

Config	ϵ	Mean	Median	Max	Over-Arch NN
Clustering 1	∞	+1.27%	-0.23%	-0.10%	**+1.86%**
	3.78	+0.63%	-0.14%	+0.11%	**+0.66%**
	1.56	+0.32%	-0.23%	+0.34%	**+0.68%**
Clustering 2	∞	+1.92%	-0.46%	-1.64%	**+2.29%**
	3.78	+0.75%	-0.21%	+0.03%	**+1.38%**
	1.56	+0.69%	-0.11%	+0.74%	**+0.76%**
Clustering 3	∞	+1.86%	+0.74%	-2.07%	**+2.31%**
	3.78	+1.49%	-0.09%	+1.03%	**+1.93%**
	1.56	+0.71%	-0.37%	+1.26%	**+1.02%**

cluster and, here, the number of partitions can be treated as a hyperparameter [21].

Effects of Features Used in Clustering: We also varied the clustering methods for each dataset and observed the effect on the final accuracy. We explored different clustering methods for different datasets and presented the best performing methods. Table 16 (Section 4.3) summarizes the clustering methods. Here, we show the result for the best performing over-arch NN-based ensemble aggregation for brevity. For the production dataset, we used user age, the app category where the ad was displayed, location (larger geographic regions), and previous click ratio of the users to cluster the users. For Taobao, we used user age, city level, and consumption level. For CelebA, we clustered the 40 binary attributes of each user using K-means clustering or simply used the number of present attributes.

Figure 2 (right) shows that clustering can affect the final model accuracy significantly. For the production dataset, clustering using the click ratio (Clustering 4) showed the best accuracy. For Taobao, clustering with city level showed the best accuracy (Clustering 3). For CelebA, using K-means clustering was the best (Clustering 3). The results show that clustering methods as well as the number of clusters are two important hyperparameters of FEL.

4.6 Evaluation Results with Differential Privacy

Table 18 shows the accuracy improvement of FEL compared to vanilla FL for two different levels of DP noise, along with the case of no DP noise ($\epsilon = \infty$). We assume the over-arch NN layer was trained with opt-in data and no DP noise added when training the over-arch NN layer. Table 18 shows that even when DP noise is added, FEL shows meaningful accuracy improvement over vanilla FL. Again, we observe that the over-arch NN layer and mean aggregations still provide the most significant gains. However, smaller ϵ leads to reduced accuracy gain, possibly due to larger injected noise. Another interesting observation is that the max ensemble aggregation improves the accuracy when DP noise is added, unlike the no-DP-noise case where it did not show any improvement. One possible reason is that DP noise mitigates the effects of outliers in training.

5 Related Work

Ensemble Distillation. Lin et al. [26] relied on unlabeled data generated by a generative model to aggregate knowledge from all heterogeneous client models, Gong et al. [14] focused on communication efficiency and privacy guarantee with one-shot offline knowledge distillation.

Boosted Federated Learning. Boosting and bagging are two prominent approaches for model ensemble learning. In Hamer et al. [16], an ensemble of pre-trained based predictors is fine tuned via federated learning, thus saving on communication costs. Luo et al. [31] suggest gradient boosting decision tree (GBDT) method, which takes the average gradient of similar samples and its own gradient as a new gradient to improve the accuracy of the local model.

Local Ensemble Learning. FedEnsemble uses random permutations to update a group of K models, and then obtains predictions through model averaging, instead of aggregating local models to update a single global model [47]. Attota et al. [3] propose a multi-view ensemble learning approach aimed at maximizing the learning efficiency of different classes of attacks for intrusion detection tasks.

Ensemble Aggregation. FedBE takes a Bayesian inference perspective by sampling and combining higher-quality global models via Bayesian ensemble for robust aggregation [7]. FedGRU uses both secure parameter aggregation and cluster ensembles to scale [28]. Orhobor et al. [41] assigned users into pre-specified bins and trained different regressors on each bin, which were later ensembled.

Although these methods have their own merits, they do not address the problem of the recommender and ranking systems use cases, in which each user has only a small number of examples, and require user-level privacy guarantee. As a result, none of these studies leverages the variation across users and diversity of behavior in their proposals. Our approach trains models on separate user clusters, leveraging a large user base in recommender and ranking systems. Furthermore, our over-arch model approach provides extra gains both in terms of precision and privacy budget.

6 Conclusion

While Federated learning (FL) has achieved considerable success as a privacy-preserving solution for model training, its impact on ranking and recommendation systems, particularly in the context of digital advertising, remains limited. We introduce Federated Ensemble Learning (FEL) to increases the learning capacity of FL. FEL can be trained efficiently without introducing significant privacy concerns and can improve the prediction accuracy meaningfully compared to vanilla FL. This work demonstrates that FEL enables FL for demanding ranking and recommendation tasks. As future work, we plan to integrate unsupervised clustering approaches, so that the segmentation can happen automatically to optimize FEL's learning performance. Finally, to minimize the cost of managing a number of leaf models, we plan to explore ways to automatically assess the quality of leaf models to pinpoint under-represented clusters and seek possible mitigation such as dynamic clustering and leaf retraining.

References

[1] S. AbdulRahman, H. Tout, H. Ould-Slimane, A. Mourad, C. Talhi, and M. Guizani. A survey on federated learning: The journey from centralized to distributed on-site learning and beyond. IEEE Internet of Things Journal, 8(7):5476–5497, 2020.

[2] B. Acun, M. Murphy, X. Wang, J. Nie, C.-J. Wu, and K. Hazelwood. Understanding training efficiency of deep learning recommendation models at scale. In 2021 IEEE International Symposium on High Performance Computer Architecture (HPCA), 2021.

[3] D. C. Attota, V. Mothukuri, R. M. Parizi, and S. Pouriyeh. An ensemble multi-view federated learning intrusion detection for IoT. IEEE Access, 9:117734–117745, 2021.

[4] T. Beane and D. Ennis. Market segmentation: a review. European journal of marketing, 1987.

[5] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth. Practical secure aggregation for privacy-preserving machine learning. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, page 1175–1191, 2017.

[6] S. Caldas, J. Konečny, H. B. McMahan, and A. Talwalkar. Expanding the reach of federated learning by reducing client resource requirements. arXiv preprint arXiv:1812.07210, 2018.

[7] H.-Y. Chen and W.-L. Chao. FedBE: Making Bayesian model ensemble applicable to federated learning. arXiv preprint arXiv:2009.01974, 2020.

[8] H.-T. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, M. Ispir, et al. Wide & deep learning for recommender systems. In Proceedings of the 1st workshop on deep learning for recommender systems, pages 7–10, 2016.

[9] E. Diao, J. Ding, and V. Tarokh. HeteroFL: Computation and communication efficient federated learning for heterogeneous clients. In 9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021. OpenReview.net, 2021.

[10] C. Dwork and J. Lei. Differential privacy and robust statistics. In Proceedings of the forty-first annual ACM symposium on Theory of computing, pages 371–380, 2009.

[11] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In Theory of Cryptography: Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006. Proceedings 3, pages 265–284. Springer, 2006.

[12] R. C. Geyer, T. Klein, and M. Nabi. Differentially private federated learning: A client level perspective. CoRR, abs/1712.07557, 2017.

[13] B. Ghazi, N. Golowich, R. Kumar, P. Manurangsi, and C. Zhang. Deep learning with label differential privacy. In Advances in Neural Information Processing Systems (NeurIPS), 2021.

[14] X. Gong, A. Sharma, S. Karanam, Z. Wu, T. Chen, D. Doermann, and A. Innanje. Preserving privacy in federated learning with ensemble cross-domain knowledge distillation. 2022.

[15] Grouplens. MovieLens 20M dataset, 2016.

[16] J. Hamer, M. Mohri, and A. T. Suresh. FedBoost: A communication-efficient algorithm for federated learning. In International Conference on Machine Learning, pages 3973–3983. PMLR, 2020.

[17] K. Hao. How Apple personalizes Siri without hoovering up your data. Technology Review, 2020.

[18] C. He, M. Annavaram, and S. Avestimehr. Group knowledge transfer: Federated learning of large CNNs at the edge. Advances in Neural Information Processing Systems, 33:14068–14080, 2020.

[19] S. Horvath, S. Laskaridis, M. Almeida, I. Leontiadis, S. Venieris, and N. Lane. FjORD: Fair and accurate federated learning under heterogeneous targets with ordered dropout. Advances in Neural Information Processing Systems, 34, 2021.

[20] D. Huba, J. Nguyen, K. Malik, R. Zhu, M. Rabbat, A. Yousefpour, C.-J. Wu, H. Zhan, P. Ustinov, H. Srinivas, et al. Papaya: Practical, private, and scalable federated learning. Proceedings of Machine Learning and Systems, 4, 2022.

[21] Y. G. Kim and C.-J. Wu. AutoFL: Enabling heterogeneity-aware energy efficient federated learning. In MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture, MICRO '21, 2021.

[22] N. K. Le, Y. Liu, Q. M. Nguyen, Q. Liu, F. Liu, Q. Cai, and S. Hirche. FedXGBoost: Privacy-preserving XGBoost for federated learning. arXiv preprint arXiv:2106.10662, 2021. Presented at International Workshop on Federated and Transfer Learning for Data Sparsity and Confidentiality (FTL-IJCAI'21).

[23] D. Li and J. Wang. FedMD: Heterogenous federated learning via model distillation. arXiv preprint arXiv:1910.03581, 2019.

[24] L. Li, J. Hong, S. Min, and Y. Xue. A novel CTR prediction model based on DeepFM for Taobao data. In 2021 IEEE International Conference on Artificial Intelligence and Industrial Design (AIID), pages 184–187. IEEE, 2021.

[25] Y. Li, Y. Zhou, A. Jolfaei, D. Yu, G. Xu, and X. Zheng. Privacy-preserving federated learning framework based on chained secure multiparty computing. IEEE Internet of Things Journal, 8(8):6178–6186, 2020.

[26] T. Lin, L. Kong, S. U. Stich, and M. Jaggi. Ensemble distillation for robust model fusion in federated learning. Advances in Neural Information Processing Systems, 33:2351–2363, 2020.

[27] Y. Liu, A. Huang, Y. Luo, H. Huang, Y. Liu, Y. Chen, L. Feng, T. Chen, H. Yu, and Q. Yang. FedVision: An online visual object detection platform powered by federated learning. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 34, pages 13172–13179, 2020.

[28] Y. Liu, J. James, J. Kang, D. Niyato, and S. Zhang. Privacy-preserving traffic flow prediction: A federated learning approach. IEEE Internet of Things Journal, 7(8):7751–7763, 2020.

[29] Z. Liu, P. Luo, X. Wang, and X. Tang. Deep learning face attributes in the wild. In Proceedings of the IEEE international conference on computer vision, pages 3730–3738, 2015.

[30] M. Lui, Y. Yetim, z. Özkan, Z. Zhao, S.-Y. Tsai, C.-J. Wu, and M. Hempstead. Understanding capacity-driven scale-out neural recommendation inference. In 2021 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), 2021.

[31] C. Luo, X. Chen, J. Xu, and S. Zhang. Research on privacy protection of multi source data based on improved GBDT federated ensemble method with different metrics. Physical Communication, 49:101347, 2021.

[32] K. Maeng, H. Lu, L. Melis, J. Nguyen, M. Rabbat, and C.-J. Wu. Towards fair federated recommendation learning: Characterizing the inter-dependence of system and data heterogeneity. arXiv preprint arXiv:2206.02633, 2022.

[33] M. Malek, I. Mironov, K. Prasad, I. Shilov, and F. Tramer. Antipodes of label differential privacy: PATE and ALIBI. In Advances in Neural Information Processing Systems (NeurIPS), 2021.

[34] H. B. McMahan, D. Ramage, K. Talwar, and L. Zhang. Learning differentially private recurrent language models. arXiv preprint arXiv:1710.06963, 2017.

[35] I. Mironov. Rényi differential privacy. In 2017 IEEE 30th computer security foundations symposium (CSF), pages 263–275. IEEE, 2017.

[36] A. Mondal, Y. More, R. H. Rooparaghunath, and D. Gupta. Flatee: Federated learning across trusted execution environments. arXiv preprint arXiv:2111.06867, 2021.

[37] M. Nalpas and S. Dutton. A more private way to measure ad conversions, the event conversion measurement API, Oct. 2020.

[38] M. Naumov, D. Mudigere, H.-J. M. Shi, J. Huang, N. Sundaraman, J. Park, X. Wang, U. Gupta, C.-J. Wu, A. G. Azzolini, et al. Deep learning recommendation model for personalization and recommendation systems. arXiv preprint arXiv:1906.00091, 2019.

[39] J. Nguyen, K. Malik, H. Zhan, A. Yousefpour, M. Rabbat, M. Malek, and D. Huba. Federated learning with buffered asynchronous aggregation. arXiv preprint arXiv:2106.06639, 2021.

[40] C. Niu, F. Wu, S. Tang, L. Hua, R. Jia, C. Lv, Z. Wu, and G. Chen. Billion-scale federated learning on mobile clients: A submodel design with tunable privacy. In Proceedings of the 26th Annual International Conference on Mobile Computing and Networking, pages 1–14, 2020.

[41] O. I. Orhobor, L. N. Soldatova, and R. D. King. Federated ensemble regression using classification. In International Conference on Discovery Science, pages 325–339. Springer, 2020.

[42] V. Perifanis and P. S. Efraimidis. Federated neural collaborative filtering. Knowledge-Based Systems, 242:108441, 2022.

[43] J. J. Pfeiffer III, D. Charles, D. Gilton, Y. H. Jung, M. Parsana, and E. Anderson. Masked LARk: Masked learning, aggregation and reporting workflow. arXiv preprint arXiv:2110.14794, 2021.

[44] A. M. Prasad, L. R. Iverson, and A. Liaw. Newer classification and regression tree techniques: bagging and random forests for ecological prediction. Ecosystems, 9(2):181–199, 2006.

[45] N. Rieke, J. Hancox, W. Li, F. Milletari, H. R. Roth, S. Albarqouni, S. Bakas, M. N. Galtier, B. A. Landman, K. Maier-Hein, et al. The future of digital health with federated learning. NPJ digital medicine, 3(1):1–7, 2020.

[46] P. Sabnagapati. Ad display/click data on taobao.com, 2020.

[47] N. Shi, F. Lai, R. A. Kontar, and M. Chowdhury. Fed-ensemble: Improving generalization through model ensembling in federated learning. arXiv preprint arXiv:2107.10663, 2021.

[48] S. Truex, N. Baracaldo, A. Anwar, T. Steinke, H. Ludwig, R. Zhang, and Y. Zhou. A hybrid approach to privacy-preserving federated learning - (extended abstract). Inform. Spektrum, 42(5):356–357, 2019.

[49] P. Vepakomma, O. Gupta, T. Swedish, and R. Raskar. Split learning for health: Distributed deep learning without sharing raw patient data. arXiv preprint arXiv:1812.00564, 2018.

[50] R. Wang, B. Fu, G. Fu, and M. Wang. Deep & cross network for ad click predictions. In Proceedings of the ADKDD'17, pages 1–7. 2017.

[51] K. Wei, J. Li, M. Ding, C. Ma, H. H. Yang, F. Farokhi, S. Jin, T. Q. Quek, and H. V. Poor. Federated learning with differential privacy: Algorithms and performance analysis. IEEE Transactions on Information Forensics and Security, 15:3454–3469, 2020.

[52] M. Wilkening, U. Gupta, S. Hsia, C. Trippel, C.-J. Wu, D. Brooks, and G.-Y. Wei. RecSSD: Near data processing for solid state drive based recommendation inference. In Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, 2021.

[53] W. Zhao, D. Xie, R. Jia, Y. Qian, R. Ding, M. Sun, and P. Li. Distributed hierarchical GPU parameter server for massive scale deep learning ads systems. Proceedings of Machine Learning and Systems, 2:412–428, 2020.

[54] G. Zhou, X. Zhu, C. Song, Y. Fan, H. Zhu, X. Ma, Y. Yan, J. Jin, H. Li, and K. Gai. Deep interest network for click-through rate prediction. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pages 1059–1068, 2018.

Enhance Mono-modal Sentiment Classification With Federated Cross-modal Transfer

Xueyang Wu[†] Di Jiang[‡] Yuanfeng Song[‡] Qian Xu[‡] Qiang Yang[†]
[†] Department of CSE, HKUST, Hong Kong, China {xwuba,qyang}@cse.ust.hk
[‡] AIGroup, WeBank Co. Ltd., Shenzhen, China {dijiang, yfsong, qianxu}@webank.com

Abstract

Sentiment analysis is a complex process that involves multiple modalities, which can provide more accurate and informative results than using a single modality. Although existing multimodal approaches have shown to be superior to mono-modal sentiment classification, they are not always practical in real-world scenarios where only mono-modal input is available, or where multimodal data is limited due to data scarcity or privacy concerns. To address this issue, we propose a novel approach that enhances mono-modal sentiment classification through federated transfer learning. Specifically, we focus on a practical industrial problem where text and speech data are owned by different affiliations, and we aim to bridge these modalities by sharing a cross-modal feature generator and phone classifier. Our proposed framework also incorporates differential privacy techniques to ensure privacy-preserving cross-modal transfer. Our experimental results on real-world spoken language sentiment classification corpora demonstrate the effectiveness of our proposed framework. We show that our approach can significantly improve the accuracy of mono-modal sentiment classification, even when only a limited amount of data is available.

1 Introduction

Sentiment classification has recently attracted significant interest due to its ability to automatically recognize the polarity of human emotional states or attitudes expressed in spoken or written language, which is crucial for improving the user experience in human-machine interaction. Communication among humans involves various modalities, including textual and acoustic content, facial expressions, and body gestures. However, single modality fails to capture sentimental information entirely and leads to inaccurate classification. To address this issue, researchers have proposed multi-modal sentiment or emotion classification approaches that utilize features from different modalities to enhance classification accuracy [12, 11]. Text, speech, and vision are three critical modalities for sentiment classification [29, 23], and previous research has proposed various fusion strategies to better utilize multiple modalities [33, 14]. Among these modalities, some researchers highlight the importance of text and speech modalities [20].

Despite the advantages of multi-modal sentiment classification, two obstacles hinder the application of traditional multi-modal sentiment classification in industry: *data scarcity* and *user privacy*. Collecting multi-modal data is challenging in real-world applications, and mono-modal speech and text data are often the only

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

available options in call centers or text-based conversation systems. Furthermore, annotating parallel multi-modal data is costly and laborious. Moreover, user privacy has become an increasing concern, and many authorities have enacted regulations to protect data and model privacy, such as the EU General Data Protection Regulation (GDPR) [27] in 2018, followed by the US and China.

To address these challenges, we propose a novel federated machine learning paradigm for sentiment classification. Since mono-modal sentimental corpora are more accessible and cost-effective, it is worth enhancing mono-modal sentiment classification if we can leverage multiple corpora with different modalities, leveraging the complementarity of different modalities from multiple sources. To address user privacy concerns when using multiple corpora, we introduce federated learning [30], a collaborative machine learning paradigm where multiple parties jointly learn a global model without exposing their private local data. Our work focuses on scenarios where each party in the federation owns a single-modal sentimental corpus and aims to enhance their sentiment classification performance collaboratively. Further details on related work are provided in Section 2.

Our proposed framework enables multiple institutes with different modality corpora to collaborate and transfer their knowledge learned from their data in a privacy-preserving manner. The cross-modal transfer is based on the phone sequence, which can be obtained from both text and speech modalities, embedding both phonetic cues and lexical information. Our method takes mono-modal input but utilizes cross-modal transfer to expand the number of modalities. Using the phone sequence as the intermedium between speech and text modality has three benefits: 1) the phone sequence is easy to obtain for both textual and audio input without modifying the existing model significantly; 2) as an intermedium between speech and text, it preserves semantic meaning and speech characteristics; 3) it is straightforward to align the audio pieces and words using the phone sequence at the utterance level. The alignment allows us to enrich the raw text input with speech features learned from the audio modality and embed the acoustic features with semantic meanings learned from the text. During collaborative training, speech-side and text-side institutions join as a federation, only exchanging intermediate model parameters protected by the differential-privacy mechanism instead of sharing raw data or bare model parameters.

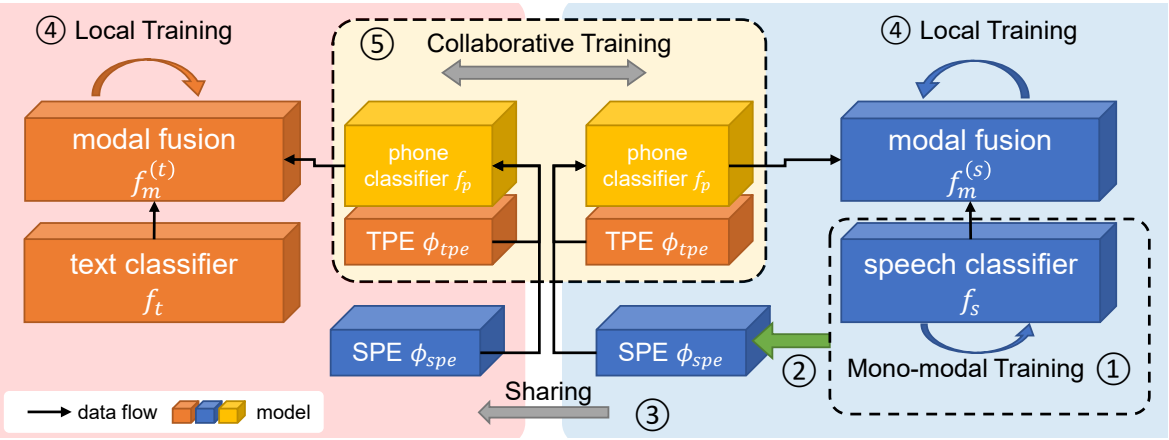


Figure 1: The Joint-view of Cross-modal Transfer Learning Framework of Text and Speech module

2 Related Work

2.1 Multi-modal Sentiment Analysis

Multi-modal sentiment and emotion analysis often involve multiple modalities, such as text, speech, and vision [29, 23]. Different approaches have been proposed to leverage the information from multiple modalities more

effectively [33, 14, 21], showing that well-designed fusion strategies significantly affect the model performance. Other research works have further explored two-modal sentiment analysis, such as visual-audio fusion [16, 29] and textual-audio fusion [12, 20]. These works typically focus on model architecture design or representation extraction. In contrast, our proposed method enhances mono-modal sentiment classification by a novel federated cross-modal transfer framework, where each party in the federation owns data in only one modality.

2.2 Cross-modal Transfer

Cross-modal transfer has been used in image classification since 2013 [25]. It allows an image classifier to classify a given image class, even if it has not been seen in the training data. Cross-modal transfer leverages the textual description of the object, forces the textual modal and visual modal embedding to map into the same space, and hence allows zero-shot inference with only a textual description of the unseen class. Existing cross-modal transfer methods for sentiment classification often focus on fusing audio and visual modalities based on the correlation between speech and facial expression [2, 6]. Our approach differs in that it focuses on the semantic complementarity between speech and spoken words. We use phonetic features as an intermediary to connect audio and textual information. For example, when in intense emotion, the pronunciation of words may have distortion, which cannot be observed in the text generated by automatic speech recognition (ASR) but can be captured in the phone sequence.

2.3 Federated Learning

Federated learning was first proposed by Google researchers in 2016 [15] as a way to learn a global model without explicitly gathering data from different clients. Federated learning has since been extended to various architectures [30, 31], such as horizontal federated learning, vertical federated learning, and federated transfer learning, with different privacy protection approaches, such as differential privacy [7], homomorphic encryption [22], and multiparty security computation [32]. Among these privacy and security protection techniques, differential privacy has emerged as a widely adapted approach for deep learning because it is practical in terms of computational efficiency. Differential privacy for deep learning is a mechanism that injects deliberately generated noise into the model during the training phase, which offers strong and robust guarantees to bound the probability of revealing raw data under a reasonable threshold. However, there is a tradeoff between high utility and high privacy protection, as injecting more noise into the model leads to less utility. To achieve higher privacy protection, we apply differential privacy to our proposed method, and to maintain high utility, the noises are only injected into a small portion of the model parameters that are shared across parties. Our proposed framework is well-suited for federated learning as it allows two parties to collaborate and share knowledge while keeping sensitive data locally, which effectively relieves the challenges of data scarcity and data privacy in sentiment classification.

3 Federated Cross-modal Transfer

3.1 Settings and Notations

For simplification, we describe our setting as consisting of two parties, and we can easily extend it to the multi-party scenario. We summarize our cross-modal transfer framework in the form of the flow chart in Fig. 1, where the red block stands for text modality and the blue one for speech modality. The circled numbers state the general steps of cross-modal joint learning.

Denote datasets for text-side and speech-side parties as $\mathcal{D}_t = \{\mathbf{w}_i, y_i\}_{i=1}^{N_t}$ and $\mathcal{D}_s = \{\mathbf{s}_i, y_i\}_{i=1}^{N_s}$, respectively, consisting of pairs of the text $\mathbf{w}$ or speech $\mathbf{s}$ and the label y, where N_t and N_s are the numbers of training instances of the text-side party and the speech-side party. We use subscripts $(\cdot)_s$, $(\cdot)_t$ to distinguish where a

variable belongs to. As a decentralized federated learning setting, this framework does not involve a central orchestrator, two parties directly set up connections and exchange parameters with each other.

3.2 Phone as Intermedium

We use phone-level information as an intermedium between text and speech modalities in this work. Phones are the units of speech that constitutes the pronunciation of words. The phone sequence captures the speech pronunciation of a sentence or a word, including the tones and pronunciation variation, such as stress and rhythm, which implies a speaker's sentiment. Phonetic cues have been used to enhance multi-lingual text classification [24] and text representation learning [19], as phones entail more semantic and sentiment cues than text. On the other hand, the phone sequence is obtainable from both speech and text sides, making it a good bridge to connect the two modalities. For example, a classic DNN-HMM hybrid ASR system can conduct the `speech-to-phone` transformation to produce the phone sequence of a given speech as the byproduct. The text can also be transformed into a phone sequence with the lexicon representing the "standard" pronunciation of the text, denoting this process as `text-to-phone`.

3.3 Federated Cross-modal Transfer Framework

As shown in Fig. 1, our federated cross-modal transfer framework involves two parties and can be summarized in 5 steps. For clarity, we denote the speech-side (blue block) as client-s, and the text-side (red block) as client-t. Client-s and client-t are two independent institutes that collaboratively enhance their mono-modal sentiment classifier with federated cross-modal transfer with each following the Alg. 5 and 6 respectively.

Algorithm 4 Federated cross-modal transfer

Require: two parties client-s and client-t with speech sentiment corpus and text sentiment corpus respectively;

1: client-s conducts lines 1-5 in Alg. 5 for initialization;
2: client-t conducts lines 1-2 in Alg. 6 for initialization;
3: **while not** reach maximum rounds **do**
4: client-s conducts lines 7-12 in Alg. 5 and client-t conducts lines 4-9 in Alg. 6 **parallelly**;
5: client-s sends $\phi_{tpe}^{(s)}, f_p^{(s)}, f_m^{(s)}$ to client-t;
6: client-t sends $\phi_{tpe}^{(t)}, f_p^{(t)}, f_m^{(t)}$ to client-s;
7: client-s conducts lines 13 to update $\phi_{tpe}^{(s)}, f_p^{(s)}, f_m^{(s)}$;
8: client-t conducts lines 10 to update $\phi_{tpe}^{(t)}, f_p^{(t)}, f_m^{(t)}$;

Step 1. The workflow of the proposed framework starts with client-s training a mono-modal speech sentiment classifier. Then client-s build a Speech-Phone-Extractor (SPE) that embeds the speech signal-level information to phonemes. The speech classifier (f_s) and text classifier (f_t) are implemented with mono-modal models mentioned in Section 4.4.

Step 2. To avoid missing the focus on our framework, we propose a vanilla SPE practice, which represents a phone with its responding f_s averaged by utterances where this appears.

Step 3. Sharing SPE helps client-t build a homogeneous phone classifier as client-s, which allows two clients to start federated learning, even though their input modalities are different.

Step 4-5. Client-s and client-t have different input modalities and mono-modal models while sharing the same phone classifier and the modal-fusion weights. The input of client-s is a piece of speech features $\mathbf{x}$, noted as $\mathbf{s} = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_T]$, extracted from the raw audio wave, where T is the number of frames. The input of client-t is a sentence, i.e., a word sequence $\mathbf{w} = [w_1, w_2, ..., w_L]$, where L is the length of the sentence. We hence obtains

phone sequences from $\mathbf{s}$ and $\mathbf{w}$ with

$$\begin{aligned} \mathbf{p}_s &= \texttt{speech-to-phone}(\mathbf{s}) \ , \\ \mathbf{p}_t &= \texttt{text-to-phone}(\mathbf{w}). \end{aligned} \tag{99}$$

Thereafter, each client has two input sequences: speech/text sequence and phone sequence. The speech/text sequence is fed into the mono-modal model, i.e,

$$\mathbf{o}_s = f_s(\mathbf{s}) \ , \ \ \mathbf{o}_t = f_t(\mathbf{w}). \tag{100}$$

Processing phone sequence is identical for the two clients, so we ignore the superscript of the models $\phi_{tpe}^{(\cdot)}, f_p^{(\cdot)}, f_m^{(\cdot)}$. We extract the distributed speech-phone representation $\mathbf{r}_{sp}$ and text-phone representation $\mathbf{r}_{tp}$ of the given phone sequence $\mathbf{p}$,

$$\mathbf{r}_{sp} = \phi_{spe}(\mathbf{p}) \ , \ \ \mathbf{r}_{tp} = \phi_{tpe}(\mathbf{p}). \tag{101}$$

The phone representations from two modalities are concatenated and fed into the phone classifier, i.e.,

$$\mathbf{r}_p = [\mathbf{r}_{sp}; \mathbf{r}_{tp}] \ , \ \ \mathbf{o}_p = h_p(\mathbf{r}_p) \tag{102}$$

We apply the late fusion (a.k.a., decision-level fusion) strategy [33] to combine information from different modalities, i.e,

$$y = f_m(\mathbf{o}_s, \mathbf{o}_p), \ \ y = f_m(\mathbf{o}_t, \mathbf{o}_p), \tag{103}$$

We compute the cross-entropy loss with the output y and label l as well as the gradients of parameters through Eq. (99)-(103). The training data of text and speech are store at each client isolatedly, and the two clients only share $\phi_{tpe}^{(\cdot)}, f_p^{(\cdot)}$, and $f_m^{(\cdot)}$ to each other. $\phi_{tpe}^{(\cdot)}$ represents $\phi_{tpe}^{(s)}$ and $\phi_{tpe}^{(t)}$, for simplicity, as well as $f_p^{(\cdot)}$, and $f_m^{(\cdot)}$. All models are updated with the Stochastic Gradient Descent optimization algorithm (SGD). To achieve differentially private models, the shared models are updated with the differentially private version of SGD that provides privacy protection of the released model [1], noted as DP-SGD. In more detail, we inject Gaussian noise [8] to the parameters during optimization. The scale of noise is related to the privacy budget (ϵ, δ) indicating the probabilities of leaking privacy. A larger privacy budget allows less model perturbation.

Each client conducts local training using their local data, and conducts weighted average over parameters of $\phi_{tpe}^{(\cdot)}, f_p^{(\cdot)}$, and $f_m^{(\cdot)}$. As a decentralized federated learning scheme, two parties directly exchange differentially private parameters of shared models with each other without the need for a central server [31]. Within each party, the weighted averaging follows Eq. (104), i.e.,

$$\begin{aligned} \phi_{tpe}^{(s)} = \phi_{tpe}^{(t)} &= \frac{\left(|\mathcal{D}_s| \cdot \phi_{tpe}^{(s)} + |\mathcal{D}_t| \cdot \phi_{tpe}^{(t)}\right)}{(|\mathcal{D}_s| + |\mathcal{D}_t|)}, \\ f_p^{(s)} = f_p^{(t)} &= \frac{\left(|\mathcal{D}_s| \cdot {f_p}^{(s)} + |\mathcal{D}_t| \cdot {f_p}^{(t)}\right)}{(|\mathcal{D}_s| + |\mathcal{D}_t|)}. \end{aligned} \tag{104}$$

4 Experiments

4.1 Experimental Setup

In this study, we assess the performance of our proposed framework on two public multi-modal datasets, namely MOSI [34] and MOSEI [35], which are available in the CMU Multimodal Data SDK [1].

[1] https://github.com/A2Zadeh/CMU-MultimodalDataSDK

Algorithm 5 DP cross-modal transfer (**client-s**)

Require: local training data $\mathcal{D}_s$;

initialize the local speech classifier (f_s);
train f_s with local training data;
build SPE $\phi_{spe}^{(s)}$ with f_s according to Alg. 7;
share $\phi_{spe}^{(s)}$ to client-t;
initialize TPE $\phi_{tpe}^{(s)}$, phone-classifier $f_p^{(s)}$, and model-fusion function $f_m^{(s)}$;
while not reach maximum rounds **do**
    for local training iteration i **do**
        sample $(\mathbf{s}, \mathbf{l})$ from local training data;
        compute gradients of f_s, $\phi_{tpe}^{(s)}$, $f_p^{(s)}$, and $f_m^{(s)}$ with $(\mathbf{s}, \mathbf{l})$ according to Eq. (99)-(103);
        update parameters of f_s with `SGD`;
        update parameters of $\phi_{tpe}^{(s)}$, $f_p^{(s)}$, and $f_m^{(s)}$ with `DP-SGD`;
    send differentially private parameters of $\phi_{tpe}^{(s)}$ and $f_p^{(s)}$ to client-t;

    receive differentially private parameters of $\phi_{tpe}^{(t)}$ and $f_p^{(t)}$ from client-t;

    update $\phi_{tpe}^{(s)}$ and $f_p^{(s)}$ according to Eq. (104);
return $\phi_{spe}^{(s)}$, $\phi_{tpe}^{(s)}$, $f_p^{(s)}$, and $f_m^{(s)}$;

Algorithm 6 DP cross-modal transfer (**client-t**)

Require: local training data $\mathcal{D}_t$;

receive $\phi_{spe}^{(s)}$ from client-t;
initialize $\phi_{tpe}^{(t)}$, $f_p^{(t)}$, $f_m^{(t)}$, and text classifier (f_t);
while not reach maximum rounds **do**
    for local training iteration i **do**
        sample $(\mathbf{w}, \mathbf{l})$ from local training data ;
        compute gradients of f_t, $\phi_{tpe}^{(t)}$, $f_p^{(t)}$, and $f_m^{(t)}$ with $(\mathbf{w}, \mathbf{l})$ according to Eq. (99)-(103);
        update parameters of f_t with `SGD`;
        update parameters of $\phi_{tpe}^{(t)}$, $f_p^{(t)}$, and $f_m^{(t)}$ with `DP-SGD`;
    send differentially private parameters of $\phi_{tpe}^{(t)}$ and $f_p^{(t)}$ to client-s;

    receive differentially private parameters of $\phi_{tpe}^{(s)}$ and $f_p^{(s)}$ from client-s;

    update $\phi_{tpe}^{(t)}$ and $f_p^{(t)}$ according to Eq. (104);
return $\phi_{tpe}^{(t)}$, $f_p^{(t)}$, and $f_m^{(t)}$;

4.2 Dataset Description

Table 19 presents a summary of the datasets, including their size and label distributions. For both datasets, we leverage the training set to train the model and the validation set to fine-tune the hyperparameters. The evaluation of the models is performed on the test set using the hyperparameters chosen through the validation set.

4.3 Baseline Models and Evaluation Metrics

To demonstrate the efficacy of our framework, we compare it with two classes of representative mono-modal sentiment classification methods, i.e., Textual Modal Model and Audio Modal Model, referred to as **baseline** models. We also perform experiments in the centralized **oracle** settings. To control for variables, the oracle settings employ the neural network model proposed in this study, but its inputs (text and speech) are aligned per utterance. The oracle settings are ideal but impractical, and they indicate the model's full potential. To further validate the compatibility of our framework with state-of-the-art models, such as BERT [5] and Transformer, we substitute the mono-modal text model with pre-trained `BERTLarge` and the speech model with a standard Transformer [26].

All models are tuned with the validation set, and the evaluation metrics are **F1-scores** reported on the testing sets, which includes four testing sets obtained from two modalities and two datasets.

Table 19: The statistics of the reference label distribution

Dataset	MOSI	MOSEI
Train	**1283** 605:678‡	**16331** 8279:8052
Valid	**299** 105:124	**1871** 939:932
Test	**686** 409:277	**5057** 2375:2287

‡ positive-negative count of reference labels

4.4 Implementation

We provide details of the implementation of our proposed framework, specifically focusing on the phone feature extractor and the acoustic phone feature mapper.

The textual phone feature extractor (ϕ_{tp}) leverages the Forward-Maximum-Matching (FMM) algorithm to translate an utterance $\mathbf{w}$ to the corresponding phone sequence $\mathbf{p}_t$, with the help of a lexicon.

To recover the acoustic feature from the phone sequence, we propose a concise acoustic phone feature mapper (ϕ_{sp}). We build a mapper from each context-free phone to an acoustic feature, either from MFCCs or openSMILE [9], following Algorithm 7. For a given phone sequence $\mathbf{p}_s$, we generate a sequence of acoustic feature vectors $\mathbf{R}_s$. We obtain an utterance vector by taking the mean over $\mathbf{R}_s$.

As we focus on the sentiment classification task, the sentiment labels in the datasets are normalized to positive (intensity > 0) and negative (intensity ≤ 0). For the acoustic model, we use a handcrafted feature set extracted by openSMILE[2] [9]. This produces a set of features that indicate intensity, loudness, Mel-frequency cepstral coefficients (MFCCs), and pitch. openSMILE extracts 384-dimensional acoustic features for every 100ms-frame. The maximum sequence lengths for words, phones, and acoustic features are all limited to 800.

We have implemented the Textual Modal Model using a multi-layer bi-directional GRU model [4], referred to as **BiGRU**. For the Audio Modal Model, we have used a Convolutional Neural Network [10] (**CNN**). Furthermore, to ensure a fair comparison between different settings, we have limited the training iteration at each epoch to 1000.

[2]We used the IS09 configuration from `https://github.com/naxingyu/opensmile/blob/master/config/`.

Algorithm 7 Build speech-phone-extractor

Require: an utterance-level acoustic feature extractor M_s for speech classification, and a set of speech $\mathbf{X}_s$, a vocabulary of phone V

$\mathbf{P}_s = \texttt{speech-to-phone}(\mathbf{X}_s)$;
for each phone p in V **do**
    $\mathbf{P}_{sp} = \{\mathbf{p}_s | \ p \in \mathbf{p}_s, \ \forall \mathbf{p}_s \in \mathbf{P}_s\}$;
    $\mathbf{R}_{sp} = \{M_s(\mathbf{p}_s) | \ \mathbf{p}_s \in \mathbf{P}_{sp}\}$;
    $\mathbf{r}_{sp} = \frac{\sum_{\mathbf{r} \in \mathbf{R}_{sp}}(\mathbf{r})}{|\mathbf{R}_{sp}|}$;
    $map[p] = \mathbf{r}_{sp}$;
$\phi_{spe}(\mathbf{p}) := (map[p_j] \mid p = [p_1, ..., p_j, ..., p_N])$; **return** ϕ_{spe}

The proposed framework and baselines have been implemented using PyTorch [18], and the differential private algorithm has been applied using PyVacy [28]. We have used the differentially private optimizer [1] to train the shared phone classifier. In the differential privacy settings, we have set all δ to 10^{-5}, and we have varied ε to evaluate both protection and performance. All experiments have been conducted on a machine equipped with an Intel(R) Xeon(R) CPU E5-2630, 128 GB RAM, and 4 NVIDIA GeForce GTX TiTian XP GPUs.

4.5 Experimental Results

Table 20: The F1-score of cross-modal transfer under different privacy budgets

Dataset	Baseline (Mono-modal)	Oracle (Centralized)	$\epsilon = 1$	$\epsilon = 5$	$\epsilon = 10$	$\epsilon = 15$	$\epsilon = \infty$	Avg. #turn
MOSI-MOSI	55.61	59.23	**57.59**	57.56	57.53	57.50	56.77	3
MOSEI-MOSEI	62.02	66.91	63.54	**65.84**	65.82	65.71	65.73	4.8
MOSI-MOSEI	55.61	N/A	57.65	57.53	**58.04**	58.01	57.88	7.2
MOSEI-MOSI	62.02	N/A	65.86	65.56	**65.88**	65.86	65.58	5.8
MOSI-MOSI	57.42	65.23	**63.32**	61.59	61.33	60.81	61.97	4.2
MOSEI-MOSEI	68.53	70.91	70.30	**70.54**	70.51	69.98	69.73	5
MOSI-MOSEI	68.53	N/A	69.74	**70.04**	69.95	69.73	69.76	6.8
MOSEI-MOSI	57.42	N/A	62.01	**62.86**	62.31	60.16	60.38	6.8

Table 20 presents the performance of four data settings with varying privacy budgets. The first column provides the details of the four settings in the form of *speech-side* and *text-side* data. In addition to transferring between identical data distributions (MOSI-MOSI and MOSEI-MOSEI), we also validate our framework's effectiveness for non-IID and imbalanced distributions (MOSEI-MOSI and MOSI-MOSEI), which are more realistic in real-world scenarios [3]. The columns from $\varepsilon = 1$ to $\varepsilon = \infty$ display the results under different parameter settings. The larger the value of ϵ, the weaker the privacy protection, and the less noise injected into the framework. According to [17], ε between 6 and 14 is practical in real-world applications. When $\epsilon = \infty$, the model is trained without any differential privacy protection (e.g., `FedAvg`).

The results in Table 20 show that our federated cross-modal transfer framework enhances performance in all settings. Moreover, we observe that our framework effectively improves the performance of mono-modal inputs to reach the ideal level of aligned multimodal inputs, as compared to the oracle settings. For instance, under-setting $\epsilon = 5$, our framework achieves absolute accuracy improvements of 3.82% and 2.01% over the baselines in

[3] Note that the training data of speech-side and text-side are neither parallel nor aligned

MOSEI and MOSE datasets, respectively. The results also suggest that the improvement is more significant for the side with less training data (as shown in the MOSI-MOSI and MOSEI-MOSI results from Table 20). The numbers in the last column demonstrate that our framework converges within a few rounds, indicating low network overload. Furthermore, we find that stricter privacy protection does not necessarily lead to performance decreases. In fact, smaller privacy budgets may provide excellent performance, where the injected noise acts as regularization for the model, avoiding overfitting. Importantly, our proposed framework is fully compatible with secure-enhanced schemes such as homomorphic encryption [13] and secure multi-party computing [3].

Table 21: F1-score of large-scale models under different privacy budgets

Dataset	Baseline (Mono-modal)	Oracle (Centralized)	$\epsilon = 1$	$\epsilon = 5$	$\epsilon = 10$	$\epsilon = 15$	$\epsilon = \infty$	Avg. #turn
MOSI-MOSI	56.98	66.13	60.77	**60.96**	60.32	59.53	57.17	2
MOSEI-MOSEI	62.01	70.22	66.14	**66.81**	66.22	66.44	66.73	4.4
MOSI-MOSEI	56.98	N/A	60.12	60.33	**61.11**	60.01	59.88	6
MOSEI-MOSI	62.01	N/A	67.16	**67.86**	66.28	65.98	65.38	4.6
MOSI-MOSI	72.26	75.32	**73.12**	72.99	72.53	71.81	70.97	4.2
MOSEI-MOSEI	74.20	78.38	76.60	**77.45**	77.23	76.89	76.73	8
MOSI-MOSEI	74.20	N/A	76.47	**77.23**	76.95	76.83	76.76	10.8
MOSEI-MOSI	72.26	N/A	73.20	**74.16**	73.93	73.71	73.88	6.8

In order to evaluate the potential of our proposed framework on large-scale models, we replaced the simple CNN speech model and BiGRU text model with a standard Transformer and a pre-trained BERTLarge model, respectively. The results are presented in Table 21, which shows four settings with two testing sets under different privacy budgets, following the same format as the baseline models. Our findings are interesting. Firstly, we observed that the performance on the text-side testing set is significantly improved by using a pre-trained model. Nevertheless, our federated cross-modal transfer still helps improve the mono-modal performance, approaching the oracle performance. On the other hand, upgrading the speech-side model to a Transformer does not yield a remarkable improvement, and the performance boost provided by our framework remains consistent with the baseline models. Furthermore, we noted that when the models are larger, weaker DP protections are more likely to result in overfitting (as seen in the last few columns of Table 21).

5 Conclusions

In this paper, we have proposed a framework for privacy-preserving cross-modal sentiment classification, which leverages the power of multi-modal input features and models trained on different modalities to enhance performance. Our experimental results, both on classic and large-scale models, demonstrate the efficacy of our framework in achieving higher accuracy in sentiment classification, alleviating data scarcity issues, and preserving data privacy for all parties involved. Our work represents a promising initial exploration into knowledge transfer among private modality data, which may hold great potential for improving mono-modal performance via federated cross-modal transfer. However, we acknowledge that this approach also presents challenges, such as privacy protection and communication efficiency, which require further research. We hope that this work will inspire future investigations in this area.

References

[1] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang. Deep learning with differential privacy. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, pages 308–318. ACM, 2016.

[2] S. Albanie, A. Nagrani, A. Vedaldi, and A. Zisserman. Emotion recognition in speech using cross-modal transfer in the wild. In 2018 ACM Multimedia Conference on Multimedia Conference, MM 2018, Seoul, Republic of Korea, October 22-26, 2018, pages 292–301, 2018.

[3] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth. Practical secure aggregation for privacy-preserving machine learning. In proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, pages 1175–1191, 2017.

[4] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555, 2014.

[5] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805, 2018.

[6] S. H. Dumpala, I. Sheikh, R. Chakraborty, and S. K. Kopparapu. Audio-visual fusion for sentiment classification using cross-modal autoencoder. NIPS2018 ViGIL Workshop, 2018.

[7] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In Theory of cryptography conference, pages 265–284. Springer, 2006.

[8] C. Dwork, A. Roth, et al. The algorithmic foundations of differential privacy. Foundations and Trends® in Theoretical Computer Science, 9(3–4):211–407, 2014.

[9] F. Eyben, M. Wöllmer, and B. Schuller. Opensmile: the munich versatile and fast open-source audio feature extractor. In Proceedings of the 18th ACM international conference on Multimedia, pages 1459–1462. ACM, 2010.

[10] Y. Kim. Convolutional neural networks for sentence classification. arXiv preprint arXiv:1408.5882, 2014.

[11] A. Korhonen, D. R. Traum, and L. Màrquez, editors. Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers. Association for Computational Linguistics, 2019.

[12] B. Li, D. Dimitriadis, and A. Stolcke. Acoustic and lexical sentiment analysis for customer service calls. In ICASSP, pages 5876–5880. IEEE, 2019.

[13] Y. Liu, Y. Kang, C. Xing, T. Chen, and Q. Yang. A secure federated transfer learning framework. IEEE Intelligent Systems, 35(4):70–82, 2020.

[14] N. Majumder, D. Hazarika, A. Gelbukh, E. Cambria, and S. Poria. Multimodal sentiment analysis using hierarchical fusion with context modeling. Knowledge-Based Systems, 161:124–133, 2018.

[15] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, et al. Communication-efficient learning of deep networks from decentralized data. arXiv preprint arXiv:1602.05629, 2016.

[16] A. Metallinou, S. Lee, and S. Narayanan. Audio-visual emotion recognition using gaussian mixture models for face and voice. In 2008 Tenth IEEE International Symposium on Multimedia, pages 250–257. IEEE, 2008.

[17] A. Orr. Google's differential privacy may be better than apple's, 2017.

[18] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in PyTorch. In NIPS Autodiff Workshop, 2017.

[19] H. Peng, Y. Ma, S. Poria, Y. Li, and E. Cambria. Phonetic-enriched text representation for chinese sentiment analysis with reinforcement learning. Information Fusion, 70:88–99, 2021.

[20] S. Poria, E. Cambria, D. Hazarika, N. Majumder, A. Zadeh, and L.-P. Morency. Context-dependent sentiment analysis in user-generated videos. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 873–883, 2017.

[21] S. Poria, I. Chaturvedi, E. Cambria, and A. Hussain. Convolutional mkl based multimodal emotion recognition and sentiment analysis. In 2016 IEEE 16th international conference on data mining (ICDM), pages 439–448. IEEE, 2016.

[22] R. L. Rivest, L. Adleman, M. L. Dertouzos, et al. On data banks and privacy homomorphisms. Foundations of secure computation, 4(11):169–180, 1978.

[23] V. Rozgić, S. Ananthakrishnan, S. Saleem, R. Kumar, and R. Prasad. Ensemble of svm trees for multimodal emotion recognition. In Proceedings of The 2012 Asia Pacific Signal and Information Processing Association Annual Summit and Conference, pages 1–4. IEEE, 2012.

[24] S. K. Singh and M. K. Sachan. Classification of code-mixed bilingual phonetic text using sentiment analysis. International Journal on Semantic Web and Information Systems (IJSWIS), 17(2):59–78, 2021.

[25] R. Socher, M. Ganjoo, C. D. Manning, and A. Ng. Zero-shot learning through cross-modal transfer. In Advances in neural information processing systems, pages 935–943, 2013.

[26] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In Advances in neural information processing systems, pages 5998–6008, 2017.

[27] P. Voigt and A. Von dem Bussche. The eu general data protection regulation (gdpr). A Practical Guide, 1st Ed., Cham: Springer International Publishing, 2017.

[28] C. Waites. Pyvacy: Towards practical differential privacy for deep learning. Article, 2019.

[29] M. Wöllmer, F. Weninger, T. Knaup, B. Schuller, C. Sun, K. Sagae, and L.-P. Morency. Youtube movie reviews: Sentiment analysis in an audio-visual context. IEEE Intelligent Systems, 28(3):46–53, 2013.

[30] Q. Yang, Y. Liu, T. Chen, and Y. Tong. Federated machine learning: Concept and applications. ACM Transactions on Intelligent Systems and Technology (TIST), 10(2):1–19, 2019.

[31] Q. Yang, Y. Liu, Y. Cheng, Y. Kang, T. Chen, and H. Yu. Federated Learning. Morgan & Claypool Publishers, 2020.

[32] A. C.-C. Yao. Protocols for secure computations. In FOCS, volume 82, pages 160–164, 1982.

[33] A. Zadeh, M. Chen, S. Poria, E. Cambria, and L.-P. Morency. Tensor fusion network for multimodal sentiment analysis. arXiv preprint arXiv:1707.07250, 2017.

[34] A. Zadeh, R. Zellers, E. Pincus, and L.-P. Morency. Mosi: multimodal corpus of sentiment intensity and subjectivity analysis in online opinion videos. arXiv preprint arXiv:1606.06259, 2016.

[35] A. B. Zadeh, P. P. Liang, S. Poria, E. Cambria, and L.-P. Morency. Multimodal language analysis in the wild: Cmu-mosei dataset and interpretable dynamic fusion graph. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 2236–2246, 2018.

NVIDIA FLARE: Federated Learning from Simulation to Real-World

Holger R. Roth Yan Cheng Yuhong Wen Isaac Yang Ziyue Xu Yuan-Ting Hsieh
Kristopher Kersten Ahmed Harouni Can Zhao Kevin Lu Zhihong Zhang Wenqi Li
Andriy Myronenko Dong Yang Sean Yang Nicola Rieke Abood Quraini Chester Chen
Daguang Xu Nic Ma Prerna Dogra Mona Flores Andrew Feng

NVIDIA Corporation*
Shanghai, China
Munich, Germany
Bethesda, Santa Clara, USA

Abstract

Federated learning (FL) enables building robust and generalizable AI models by leveraging diverse datasets from multiple collaborators without centralizing the data. We created NVIDIA FLARE[1] as an open-source software development kit (SDK) to make it easier for data scientists to use FL in their research and real-world applications. The SDK includes solutions for state-of-the-art FL algorithms and federated machine learning approaches, which facilitate building workflows for distributed learning across enterprises and enable platform developers to create a secure, privacy-preserving offering for multiparty collaboration utilizing homomorphic encryption or differential privacy. The SDK is a lightweight, flexible, and scalable Python package. It allows researchers to apply their data science workflows in any training libraries (PyTorch, TensorFlow, XGBoost, or even NumPy) in real-world FL settings. This paper introduces the key design principles of NVFlare and illustrates some use cases (e.g., COVID analysis) with customizable FL workflows that implement different privacy-preserving algorithms.

1 Introduction

Federated learning (FL) has become a reality for many real-world applications [34]. It enables multinational collaborations on a global scale to build more robust and generalizable machine learning and AI models. In this paper, we introduce NVIDIA FLARE (NVFlare), an open-source software development kit (SDK) that makes it easier for data scientists to collaborate to develop more generalizable and robust AI models by sharing model weights rather than private data. While FL is attractive in many industries, it is particularly beneficial for healthcare applications where patient data needs to be protected. For example, FL has been used for predicting clinical

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

*Contact: {hroth,yanc,chesterc,daguangx,pdogra,andyf}@nvidia.com
[1]Code is available at https://github.com/NVIDIA/NVFlare.

outcomes in patients with COVID-19 [7] or to segment brain lesions in magnetic resonance imaging [39, 38]. NVFlare is not limited to applications in healthcare and is designed to allow cross-silo FL [18] across enterprises for different industries and researchers.

In recent years, several efforts (both open-source and commercial) have been made to bring FL technology into the healthcare sector and other industries, like TensorFlow Federated [1], PySyft [48], FedML [14], FATE [26], Flower [3], OpenFL [33], Fed-BioMed [40], IBM Federated Learning [27], HP Swarm Learning [42], FederatedScope [44], FLUTE [8], and more. Some focus on simulated FL settings for researchers, while others prioritize production settings. NVFlare aims to be useful for both scenarios: 1) for researchers by providing efficient and extensible simulation tools and 2) by providing an easy path to transfer research into real-world production settings, supporting high availability and server failover, and by providing additional productivity tools such as multi-tasking and admin commands.

2 NVIDIA FLARE Overview

NVIDIA FLARE – or short NVFlare – stands for "**NV**IDIA **F**ederated **L**earning **A**pplication **R**untime **E**nvironment". The SDK enables researchers and data scientists to adapt their machine learning and deep learning workflows to a federated paradigm. It enables platform developers to build a secure, privacy-preserving offering for distributed multiparty collaboration.

NVFlare is a lightweight, flexible, and scalable FL framework implemented in Python that is agnostic to the underlying training library. Developers can bring their own data science workflows implemented in PyTorch, TensorFlow, or even in pure NumPy, and apply them in a federated setting. A typical FL workflow such as the popular federated averaging (FedAvg) algorithm [28], can be implemented in NVFlare using the following main steps. Starting from an initial global model, each FL client trains the model on their local data for a while and sends model updates to the server for aggregation. The server then uses the aggregated updates to update the global model for the next round of training. This process is iterated many times until the model converges.

Though used heavily for federated deep learning, NVFlare is a generic approach for supporting collaborative computing across multiple clients. NVFlare provides the *Controller* programming API for researchers to create workflows for coordinating clients for collaboration. FedAvg is one such workflow. Another example is cyclic weight transfer [5]. The central concept of collaboration is the notion of "task". An FL controller assigns tasks (e.g., deep-learning training with model weights) to one or more FL clients and processes results returned from clients (e.g., model weight updates). The controller may assign additional tasks to clients based on the processed results and other factors (e.g., a pre-configured number of training rounds). This task-based interaction continues until the objectives of the study are achieved.

The API supports typical controller-client interaction patterns like broadcasting a task to multiple clients, sending a task to one or more specified clients, or relaying a task to multiple clients sequentially. Each interaction pattern has two flavors: wait (block until client results are received) or no-wait. A workflow developer can use these interaction patterns to create innovative workflows. For example, the *ScatterAndGather* controller (typically used for FedAvg-like algorithms) is implemented with the *broadcast_and_wait* pattern, and the *CyclicController* is implemented with the *relay_and_wait pattern*. The controller API allows the researcher to focus on the control logic without needing to deal with underlying communication issues. Figure 1 shows the principle. Each FL client acts as a worker that simply executes tasks assigned to it (e.g., model training) and returns execution results to the controller. At each task interaction, there can be optional filters that process the task data or results before passing it to the *Controller* (on the server side) or task executor (client side). The filter mechanism can be used for data privacy protection (e.g., homomorphic encryption/decryption or differential privacy) without having to alter the training algorithms.

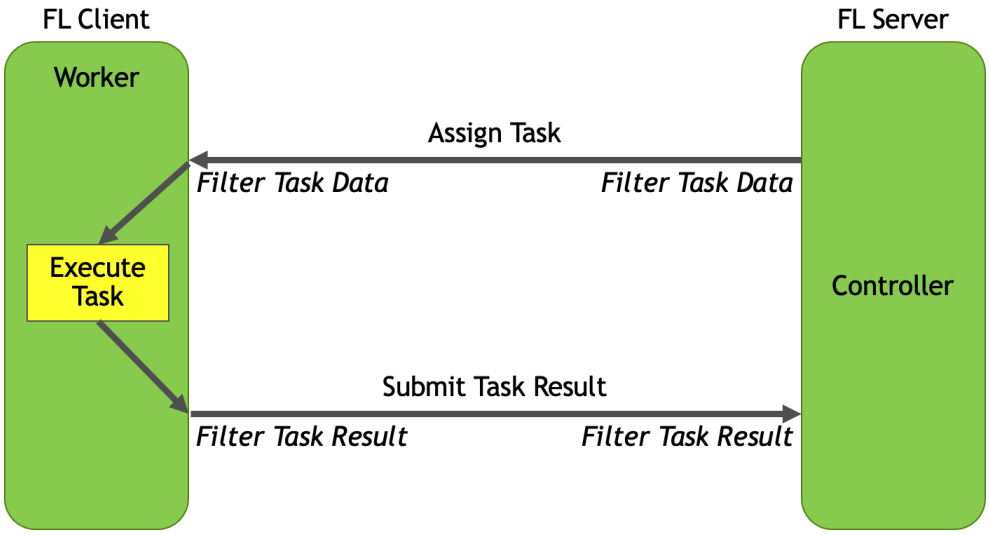


Figure 1: NVFlare job execution. The *Controller* is a Python object that controls or coordinates the *Workers* to get a job done. The controller is run on the FL server. A *Worker* is capable of performing tasks. *Workers* run on FL clients.

Key Components NVFlare is built on a componentized architecture that allows FL workloads to move from research and simulation to real-world production deployment. Some of the key components of this SDK include:

- **FL Simulator** for rapid development and prototyping.
- **NVFlare Dashboard** for simplified project management, secure provisioning, and deployment, orchestration.
- **Reference FL algorithms** (e.g., FedAvg, FedProx, SCAFFOLD) and workflows, like scatter and gather, cyclic, etc.
- **Privacy preservation** with differential privacy, homomorphic encryption, and more.
- **Specification-based API** for extensibility, allowing customization with plug-able components.
- **Tight integration** with other learning frameworks like MONAI [4], XGBoost [6], and more.

High-Level Architecture NVFlare is designed with the idea that less is more, using a specification-based design principle to focus on what is essential. This allows other people to be able to do what they want to do in real-world applications by following clear API definitions. FL is an open-ended space. The API-based design allows others to bring their implementations and solutions for various components. Controllers, task executors, and filters are just examples of such extensible components. NVFlare provides an end-to-end operation environment for different personas. It provides a comprehensive provisioning system that creates security credentials for secure communications to enable the easy and secure deployment of FL applications in the real world. It also provides an FL Simulator for running proof-of-concept studies locally. In production mode, the researcher conducts an FL study by submitting jobs using admin commands using Notebooks or the NVFlare Console – an interactive command tool. NVFlare provides many commands for system operation and job management. With these commands, one can start and stop a specific client or the entire system, submit new jobs, check the status of jobs, create a job by cloning from an existing one, and much more.

With NVFlare's component-based design, a job is just a configuration of components needed for the study. For the control logic, the job specifies the controller component to be used and any components required by the controller.

3 System Concepts

A NVFlare system is a typical client-server communication system that comprises one or more FL server(s), one or more FL client(s), and one or more admin clients. The FL Servers open two ports for communication with FL clients and admin clients. FL clients and admin clients connect to the opened ports. FL clients and admin clients do not open any ports and do not directly communicate with each other. The following is an overview of the key concepts and objects available in NVFlare and the information that can be passed between them.

Workers and Controller NVFlare's collaborative computing is achieved through the *Controller*/*Worker* interactions.

Shareable Object that represents a communication between server and client. Technically, the *Shareable* is implemented as a Python dictionary that could contain different information, e.g., model weights.

Data Exchange Object (DXO) Standardizes the data passed between the communicating parties. One can think of the *Shareable* as the envelope and the *DXO* as the letter. Together, they comprise a message to be shared between communicating parties.

FLComponent The base class of all the FL components. Executors, controllers, filters, aggregators, and their subtypes are all *FLComponents*. *FLComponent* comes with some useful built-in methods for logging, event handling, auditing, and error handling.

Executors Type of *FLComponent* for FL clients that has an execute method that produces a *Shareable* from an input *Shareable*. NVFlare provides both single- and multi-process executors to implement different computing workloads.

FLContext One of the most important features of NVFlare is to pass data between the FL components. *FLContext* is available to every method of all common FLComponent types. Through *FLContext*, the component developer can get services provided by the underlying infrastructure and share data with other components of the FL system.

Communication Drivers NVFlare abstracts the communication layers out so that different deployment scenarios can implement customizable communication drivers. By default, we use GRPC for data communication in task-based communication. However, the driver can be replaced to run other communication protocols, for example, TCP. The customizable nature of communication in NVFlare allows for both server-centric and peer-to-peer communication patterns. This enables the user to utilize both scatter and gather-type workflows like FedAvg [28], decentralized training patterns like swarm learning [42], or direct peer-to-peer communication as in split learning [12].

Fig. 2 compares the times for model upload and download from the client's perspective using different communication protocols available in NVFlare using a model of $\sim$18MB in size.

The experiment runs in a multi-cloud environment with the server and eight clients running on Azure, while two clients run on AWS. One can observe that the global model download is slower as all clients are trying to

download the global model at the same time, and hence the server is more busy. In contrast, the clients' model uploads happen at slightly different times and therefore are faster. One can also see how this multi-cloud setup causes the clients on AWS to take slightly longer during model download due to communication across different cloud infrastructures.

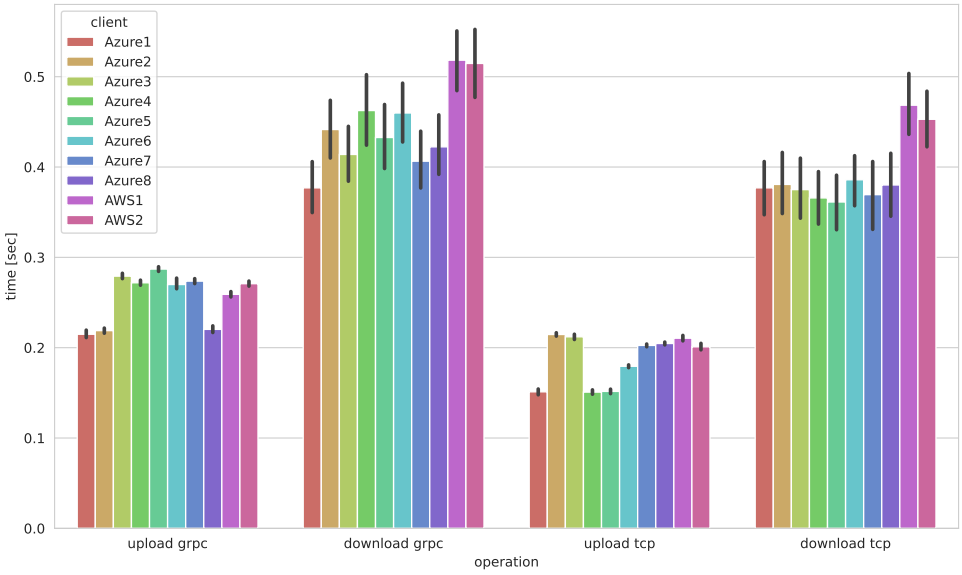


Figure 2: Comparison of GRPC and TCP communication drivers in NVFlare. The server is running on Azure. The clients are distributed between Azure and AWS. The message size is ∼18MB. Communication times were measured over 100 rounds of FedAvg. Error bars indicate the 95% confidence intervals.

Filters Filters in NVFlare are a type of *FLComponent* that have a process method to transform the *Shareable* object between the communicating parties. A Filter can provide additional processing to shareable data before sending or after receiving from a peer. Filters can convert data formats and a lot more and are NVFlare's primary mechanism for data privacy protection [24, 13]:

- *ExcludeVars* to exclude variables from shareable.
- *PercentilePrivacy* for truncation of weights by percentile.
- *SVTPrivacy* for differential privacy through sparse vector techniques.
- Homomorphic encryption filters used for secure aggregation.

As an example, we show the average encryption, decryption, and upload times when using homomorphic encryption for secure aggregation[2]. We compare raw data to encrypted model gradients uploaded in Table 22 when hosting the server on AWS[3] and connecting 30 client instances using an on-premise GPU cluster. One can see the longer upload times due to the larger message sizes needed by homomorphic encryption.

[2] `https://developer.nvidia.com/blog/federated-learning-with-homomorphic-encryption`

[3] For reference, we used an m5a.2xlarge instance with eight vCPUs, 32-GB memory, and up to 2,880 Gbps network bandwidth.

Table 22: Federated learning exchanging homomorphic encrypted vs. raw model updates.

Time in seconds	Mean	Std. Dev.
Encryption	5.01	1.18
Decryption	0.95	0.04
Enc. upload	38.00	71.17
Raw upload	21.57	74.23

Event Mechanism NVFlare comes with a powerful event mechanism that allows dynamic notifications to be sent to all event handlers. This mechanism enables data-based communication among decoupled components: one component fires an event when a certain condition occurs, and other components can listen to that event and processes the event data. Each *FLComponent* is automatically an event handler. To listen to and process an event, one can simply implement the *handle_event()* method and process desired event types. Events represent some important moments during the execution of the system logic. For example, before and after aggregation or when important data becomes available, e.g., a new "best" model was selected.

3.1 Productivity Features

NVFlare contains features that enable efficient, collaborative, and robust computing workflows.

Multi-tasking For systems with a large capacity, computing resources could be idle most of the time. NVFlare implements a resource-based multi-tasking solution, where multiple jobs can be run concurrently when overall system resources are available. Multi-tasking is made possible by a job scheduler on the server side that constantly tries to schedule a new job. For each job to be scheduled, the scheduler asks each client whether they can satisfy the required resources of the job (e.g., number of GPU devices) by querying the client's resource manager. If all clients can meet the requirement, the job will be scheduled and deployed to the clients.

High Availability and Server Failover To avoid the FL server as a single point of failure, a solution has been implemented to support multiple FL servers with automatic cut-over when the currently active server becomes unavailable. Therefore, a component called *Overseer* is added to facilitate automatic cut-over. The *Overseer* provides the authoritative endpoint info of the active FL server. All other system entities (FL servers, FL clients, admin clients) constantly communicate (i.e., every 5 seconds) with the Overseer to obtain and act on such information. If the server cutover happens during the execution of a job, then the job will continue to run on the new server. Depending on how the controller is written, the job may or may not need to restart from the beginning but can continue from a previously saved snapshot.

Simulator NVFlare provides a simulator to allow data scientists and system developers to easily write new *FLComponents* and novel workflows. The simulator is a command line tool to run a NVFlare job. To allow simple experimentation and debugging, the FL server and multiple clients run in the same process during simulation. A multi-process option allows efficient use of resources, e.g., training multiple clients on different GPUs. The simulator follows the same job execution as in real-world NVFlare deployment. Therefore, components developed in simulation can be directly deployed in real-world federated scenarios.

3.2 Secure Provisioning in NVFlare

Security is an important requirement for FL systems. NVFlare provides security solutions in the following areas: authentication, communication confidentiality, user authorization, data privacy protection, auditing, and local

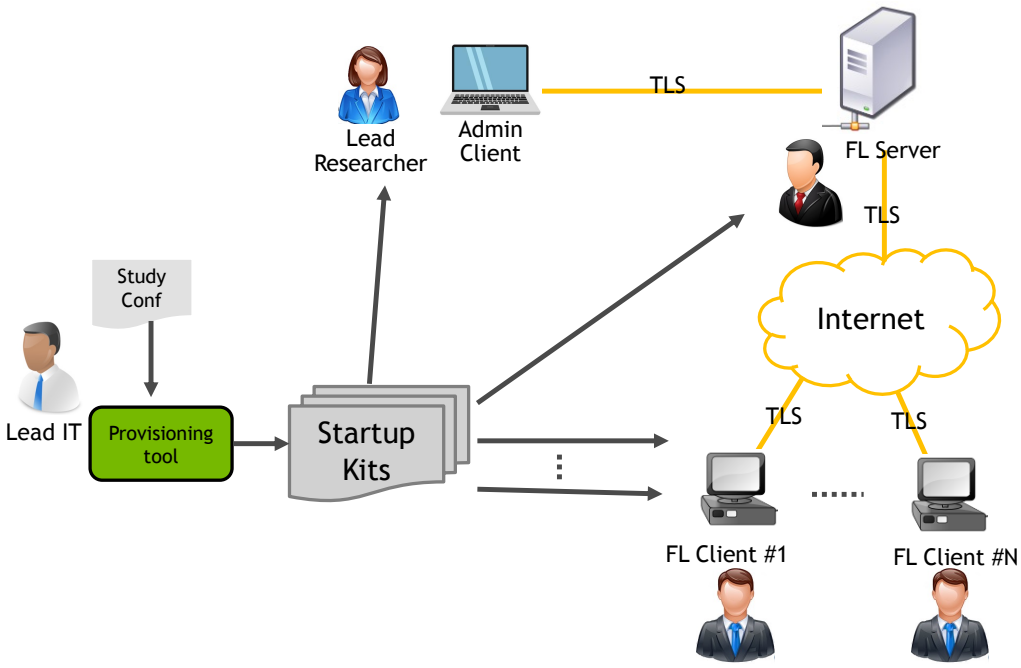


Figure 3: High-level steps for running a real-world study with secure provisioning with NVFlare.

client policies.

Authentication NVFlare ensures the identities of communicating peers using mutual Transport Layer Security (TLS). Each participating party (FL Servers, Overseer, FL Clients, Admin Clients) must be properly provisioned. Once provisioned, each party receives a startup kit containing TLS credentials (public cert of the root, the party's own private key and certificate) and system endpoint information, see Fig. 3. Each party can only connect to the NVFlare system with the startup kit. Communication confidentiality is also achieved with the use of TLS-based messaging.

Federated Authorization NVFlare's admin command system is very rich and powerful. Not every command is for everyone. NVFlare implements a role-based user authorization system that controls what a user can or cannot do. At the time of provision, each user is assigned a role. Authorization policies specify which commands are permitted for which roles. Each FL client can define its authorization policy that specifies what a role can or cannot do to the client. For example, one client could allow a role to run jobs from any researchers. In contrast, another client may only allow jobs submitted by its researchers (i.e., the client and the job submitter belong to the same organization).

NVFlare automatically records all user commands and job events in system audit files on both the server and client sides. In addition, the audit API can be used by application developers to record additional events in the audit files.

Client-Privacy NVFlare enhances the overall system security by allowing each client to define its policies for authorization, data privacy (filters), and computing resource management. The client can change its policies at any time after the system is up and running without having to be re-provisioned. For example, the client could require all jobs running on it to be subject to a set of filters. The client could also change the number of computing resources (e.g., GPU devices) to be used by the FL client.

4 Federated Data Science

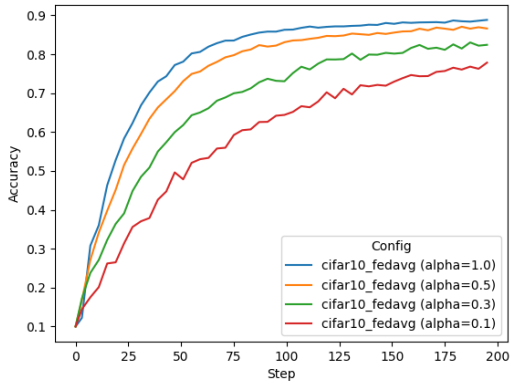
As a general distributed computing platform, NVFlare can be used for various applications in different industries. Here we describe some of the most common use cases where NVFlare was deployed.

4.1 Federated Deep Learning

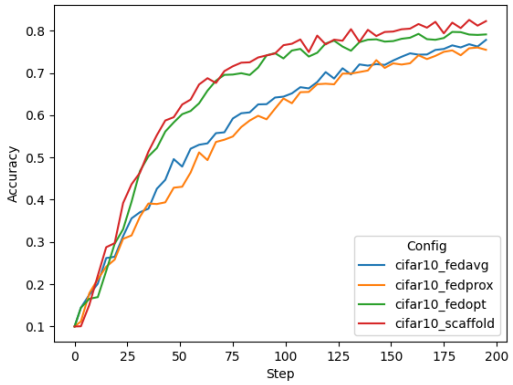
A go-to example dataset for benchmarking different FL algorithms is CIFAR-10 [20]. NVFlare allows users to experiment with different algorithms and data splits using different levels of heterogeneity based on a Dirichlet sampling strategy [41]. Figure 4(a) shows the impact of varying alpha values, where lower values cause higher heterogeneity on the performance of the FedAvg.

Apart from FedAvg, currently available in NVFlare include FedProx [23], FedOpt [32], and SCAFFOLD [19]. Figure 4(b) compares an α setting of 0.1, causing a high data heterogeneity across clients and its impact on more advanced FL algorithms, namely FedProx, FedOpt, and SCAFFOLD. FedOpt and SCAFFOLD show markedly better convergence rates and achieve better performance than FedAvg and FedProx with the same alpha setting. SCAFFOLD achieves this by adding a correction term when updating the client models, while FedOpt utilizes SGD with momentum to update the global model on the server. Therefore, both perform better with the same number of training steps as FedAvg and FedProx.

Other algorithms available in or coming soon to NVFlare include federated XGBoost [6], Ditto [22], FedSM [45], Auto-FedRL [11], and more.



(a) FedAvg with increasing levels of heterogeneity (smaller α values).



(b) FL algorithms with a heterogeneous data split (α=0.1).

Figure 4: Federated learning experiments with NVFlare.

4.2 Federated Machine Learning

Traditional machine learning methods, such as linear models, support vector machine (SVM), and k-means clustering, can be formulated under a federated setting.

With certain libraries, the federated machine learning algorithms need to be designed considering two factors: algorithm-wise, each of these models has distinct training schemes and model representations; and implementation-wise, popular libraries providing these functionalities (e.g., scikit-learn, XGBoost) have different APIs and inner logics. Hence, when developing an FL variant of a particular traditional machine learning method, several questions need to be answered at these two levels:

First, at the algorithm level, we need to break down the optimization process into individual steps/rounds (if possible) and have answers to three major questions:

1. What information should clients share with the server?
2. How should the server aggregate the collected information from clients?
3. What should clients do with the global aggregated information received from the server?

Second, at the implementation level, we need to know what APIs are available and how to utilize them in a federated pipeline to implement a distributed version of the algorithm.

A major difference between federated traditional machine learning and federated deep learning is that, for traditional machine learning methods, the boundary between "federated" and "distributed", or even "ensemble", can be much more vague than for deep learning. Due to the characteristics of a given algorithm and its API design, the concepts can be equivalent. Take XGBoost and SVM, for example: Algorithm-wise, XGBoost can distribute the training samples to several workers and construct trees based on the collected histograms from each worker. Such a process can be directly adopted under a federated setting because the communication cost is affordable. In this case, "federated" is equivalent to "distributed" learning. API-wise, some algorithms can be constrained by their implementation. Take scikit-learn's SVM for instance. Although theoretically SVM can be formulated as an iterative optimization process, the API only supports one-shot "fitting" without the capability of separately calling the optimization steps. Hence a federated SVM algorithm using the scikit-learn library can only be implemented as a two-step process. In this case, "federated" is equivalent to "ensemble".

For clarification, we provide the full formulation for tree-based federated XGBoost, illustrated in Fig. 5:

1. XGBoost, by definition, is a sequential optimization process: each step adds one extra tree to the model to reduce the residual error. Hence, federated XGBoost can be formulated as follows: each round of FL corresponds to one boosting step at the local level. Clients share the newly added tree trained on local data with the server at the end of local boosting.
2. The model representation is a decision/regression tree. To aggregate the information from all clients, the server will bag all received trees to form a "forest" to be added to the global boosting model.
3. With the updated global model from the server, each client will continue the boosting process by learning a new tree starting from the global model of the boosted forest.

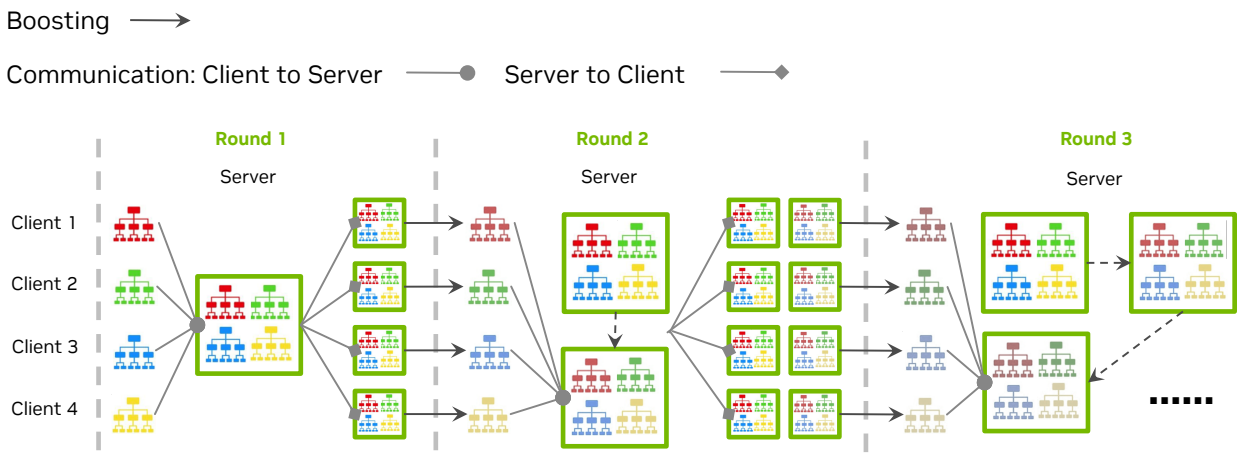
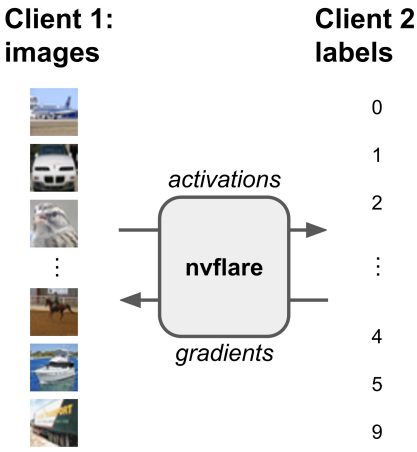


Figure 5: Tree-based federated XGBoost: a "boosting of forests."

4.3 Split learning

Split learning assumes a vertical data partitioning [46] that can be useful in many distributed learning scenarios involving neural network architectures [12].

As an introductory example, we can assume that one client holds the images, and the other holds the labels to compute losses and accuracy metrics. Activations and corresponding gradients are being exchanged between the clients using NVFlare, as illustrated in Fig. 6. We use a cryptographic technique called private set intersection



Setup	Training Time [min]
Simulated PyTorch	19
Routing through server (TCP)	27
Peer-to-peer (TCP)	25

Figure 6: Simple split learning scenario using CIFAR-10. The table compares multiple communication patterns. Using 50,000 training samples and 15,625 rounds of communication with a batch size of 64.

(PSI) [43] to compute the alignment between images and labels on both clients. NVFlare's implementation of PSI can be extended to multiple parties and applied to other use cases than split learning, e.g., requiring a secure and privacy-preserving alignment of different databases.

Using NVFlare's capability to implement different communication patterns, we can investigate the communication speed-ups one can achieve by implementing split learning using direct peer-to-peer communication as opposed to routing the messages between the two clients through a central server.

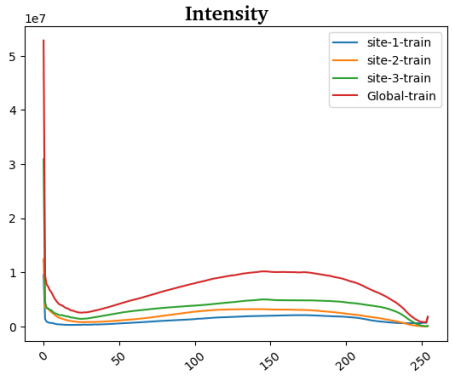
The table in Fig. 6 compares the training speeds of split learning on the CIFAR-10 dataset in a local simulation scenario. First, we use the same PyTorch script to simulate split learning. Then, we implement two distributed solutions using NVFlare. One that routes the messages through the server and one using a direct peer-to-peer connection between the clients. As expected, the direct peer-to-peer connection is more efficient, achieving only a slight overhead in total training time compared to the standalone PyTorch script, which could not be translated to real-world scenarios.

4.4 Federated Statistics

NVFlare provides built-in federated statistics operators (*Controller* and *Executors*) that will generate global statistics based on local client statistics. Each client could have one or more datasets, such as "train" and "test" datasets. Each dataset may have many features. NVFlare will calculate and combine the statistics for each feature in the dataset to produce global statistics for all the numeric features. The output gathered on the server will be the complete statistics for all datasets in clients and global, as illustrated in Fig. 7.

	count	histogram
site-2-train	6012	[[0.0, 1.0, 12430030], [1.0, 2.0, 3511491], [2...
site-4-train	1345	NaN
site-3-train	10192	[[0.0, 1.0, 30867349], [1.0, 2.0, 4553187], [2...
site-1-train	3616	[[0.0, 1.0, 9512374], [1.0, 2.0, 1381654], [2....
Global-train	21165	[[0.0, 1.0, 52809753], [1.0, 2.0, 9446332], [2...

(a) Federated statistics. Note the data of "site-4" violates the client's privacy policy and therefore does not share its statistics with the server.



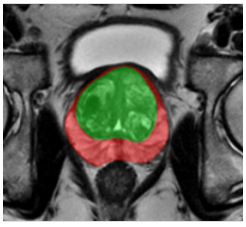
(b) Histogram visualization.

Figure 7: Federated statistics with NVFlare.

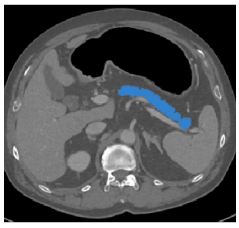
5 Real-world Use Cases

NVFlare and its predecessors have been used in several real-world studies exploring FL for healthcare scenarios. The collaborations between multinational institutions tested and validated the utility of federated learning, pushing the envelope for training robust, generalizable AI models. These initiatives included FL for breast mammography classification [35], prostate segmentation [37], pancreas segmentation [41], and most recently, chest X-ray (CXR) and electronic health record (EHR) analysis to predict the oxygen requirement for patients arriving in the emergency department with symptoms of COVID-19 [7].

(a) Mammography.



(b) Prostate.



(c) Pancreas.

(d) CXR & EHR.

Figure 8: Real-world use cases of NVFlare.

6 Summary & Conclusion

We described NVFlare, an open-source SDK to make it easier for data scientists to use FL in their research and to allow an easy transition from research to real-world deployment. As discussed above, NVFlare's *Controller* programming API supports various interaction patterns between the server and clients over internet connections, which could be unstable. Therefore, the API design mitigates various failure conditions and unexpected crashes of the client machines, such as allowing developers to process timeout conditions properly.

NVFLare's unique flexibility and agnostic approach towards the deployed training libraries make it the perfect solution for integrating with different deep learning frameworks, including popular ones used for training large language models (LLM). With our dedication to addressing the current limitations of communication protocols, we are working towards supporting the communication of large message sizes, enabling the federated

fine-tuning of AI models with billions of parameters, such as those used for ChatGPT [31] and GPT-4 [30]. Moreover, our team is implementing parameter-efficient federated methods to adapt LLM models to downstream tasks [47], utilizing techniques such as prompt tuning [21] and p-tuning [25], adapters [16, 15], LoRA [17], showing promising performance. Our commitment to innovation and excellence in this field ensures that we continue to push the boundaries of what is possible with federated learning.

We did not go into all details of exciting features available in NVFlare, like homomorphic encryption, TensorBoard streaming, provisioning web dashboard, integration with MONAI[4] [29, 4], etc. However, we hope that this overview of NVFlare gives a good starting point for developers and researchers on their journey to using FL and federated data science in simulation and the real world.

NVFlare is an open-source project. We invite the community to contribute and grow NVFlare. For more information, please visit the code repository at `https://github.com/NVIDIA/NVFlare`.

References

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. {TensorFlow}: a system for {Large-Scale} machine learning. In 12th USENIX symposium on operating systems design and implementation (OSDI 16), pages 265–283, 2016.

[2] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang. Deep learning with differential privacy. In Proceedings of the 2016 ACM SIGSAC conference on computer and communications security, pages 308–318, 2016.

[3] D. J. Beutel, T. Topal, A. Mathur, X. Qiu, T. Parcollet, P. P. de Gusmão, and N. D. Lane. Flower: A friendly federated learning research framework. arXiv preprint arXiv:2007.14390, 2020.

[4] M. J. Cardoso, W. Li, R. Brown, N. Ma, E. Kerfoot, Y. Wang, B. Murrey, A. Myronenko, C. Zhao, D. Yang, et al. Monai: An open-source framework for deep learning in healthcare. arXiv preprint arXiv:2211.02701, 2022.

[5] K. Chang, N. Balachandar, C. Lam, D. Yi, J. Brown, A. Beers, B. Rosen, D. L. Rubin, and J. Kalpathy-Cramer. Distributed deep learning networks among institutions for medical imaging. Journal of the American Medical Informatics Association, 25(8):945–954, 2018.

[6] T. Chen and C. Guestrin. XGBoost: A scalable tree boosting system. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '16, pages 785–794, New York, NY, USA, 2016. ACM.

[7] I. Dayan, H. R. Roth, A. Zhong, A. Harouni, A. Gentili, A. Z. Abidin, A. Liu, A. B. Costa, B. J. Wood, C.-S. Tsai, et al. Federated learning for predicting clinical outcomes in patients with covid-19. Nature medicine, 27(10):1735–1743, 2021.

[8] D. Dimitriadis, M. H. Garcia, D. M. Diaz, A. Manoel, and R. Sim. Flute: A scalable, extensible framework for high-performance federated learning simulations. arXiv preprint arXiv:2203.13789, 2022.

[9] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In Theory of cryptography conference, pages 265–284. Springer, 2006.

[10] J. Geiping, H. Bauermeister, H. Dröge, and M. Moeller. Inverting gradients-how easy is it to break privacy in federated learning? Advances in Neural Information Processing Systems, 33:16937–16947, 2020.

[4] `https://monai.io`

[11] P. Guo, D. Yang, A. Hatamizadeh, A. Xu, Z. Xu, W. Li, C. Zhao, D. Xu, S. Harmon, E. Turkbey, et al. Auto-fedrl: Federated hyperparameter optimization for multi-institutional medical image segmentation. arXiv preprint arXiv:2203.06338, 2022.

[12] O. Gupta and R. Raskar. Distributed learning of deep neural network over multiple agents. Journal of Network and Computer Applications, 116:1–8, 2018.

[13] A. Hatamizadeh, H. Yin, P. Molchanov, A. Myronenko, W. Li, P. Dogra, A. Feng, M. G. Flores, J. Kautz, D. Xu, et al. Do gradient inversion attacks make federated learning unsafe? arXiv preprint arXiv:2202.06924, 2022.

[14] C. He, S. Li, J. So, X. Zeng, M. Zhang, H. Wang, X. Wang, P. Vepakomma, A. Singh, H. Qiu, et al. FedML: A research library and benchmark for federated machine learning. arXiv preprint arXiv:2007.13518, 2020.

[15] J. He, C. Zhou, X. Ma, T. Berg-Kirkpatrick, and G. Neubig. Towards a unified view of parameter-efficient transfer learning. arXiv preprint arXiv:2110.04366, 2021.

[16] N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. De Laroussilhe, A. Gesmundo, M. Attariyan, and S. Gelly. Parameter-efficient transfer learning for nlp. In International Conference on Machine Learning, pages 2790–2799. PMLR, 2019.

[17] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen. Lora: Low-rank adaptation of large language models. arXiv preprint arXiv:2106.09685, 2021.

[18] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, et al. Advances and open problems in federated learning. arXiv preprint arXiv:1912.04977, 2019.

[19] S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and A. T. Suresh. Scaffold: Stochastic controlled averaging for federated learning. In International Conference on Machine Learning, pages 5132–5143. PMLR, 2020.

[20] A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[21] B. Lester, R. Al-Rfou, and N. Constant. The power of scale for parameter-efficient prompt tuning. arXiv preprint arXiv:2104.08691, 2021.

[22] T. Li, S. Hu, A. Beirami, and V. Smith. Ditto: Fair and robust federated learning through personalization. In International Conference on Machine Learning, pages 6357–6368. PMLR, 2021.

[23] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith. Federated optimization in heterogeneous networks. Proceedings of Machine Learning and Systems, 2:429–450, 2020.

[24] W. Li, F. Milletarì, D. Xu, N. Rieke, J. Hancox, W. Zhu, M. Baust, Y. Cheng, S. Ourselin, M. J. Cardoso, et al. Privacy-preserving federated brain tumour segmentation. In International workshop on machine learning in medical imaging, pages 133–141. Springer, 2019.

[25] X. Liu, Y. Zheng, Z. Du, M. Ding, Y. Qian, Z. Yang, and J. Tang. Gpt understands, too. arXiv preprint arXiv:2103.10385, 2021.

[26] Y. Liu, T. Fan, T. Chen, Q. Xu, and Q. Yang. Fate: An industrial grade platform for collaborative learning with data protection. J. Mach. Learn. Res., 22(226):1–6, 2021.

[27] H. Ludwig, N. Baracaldo, G. Thomas, Y. Zhou, A. Anwar, S. Rajamoni, Y. Ong, J. Radhakrishnan, A. Verma, M. Sinn, et al. Ibm federated learning: an enterprise framework white paper v0. 1. arXiv preprint arXiv:2007.10987, 2020.

[28] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas. Communication-efficient learning of deep networks from decentralized data. In Artificial intelligence and statistics, pages 1273–1282. PMLR, 2017.

[29] MONAI Consortium. MONAI: Medical Open Network for AI, 9 2022.

[30] OpenAI. Gpt-4 technical report, 2023.

[31] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, et al. Training language models to follow instructions with human feedback. Advances in Neural Information Processing Systems, 35:27730–27744, 2022.

[32] S. Reddi, Z. Charles, M. Zaheer, Z. Garrett, K. Rush, J. Konečnỳ, S. Kumar, and H. B. McMahan. Adaptive federated optimization. arXiv preprint arXiv:2003.00295, 2020.

[33] G. A. Reina, A. Gruzdev, P. Foley, O. Perepelkina, M. Sharma, I. Davidyuk, I. Trushkin, M. Radionov, A. Mokrov, D. Agapov, et al. Openfl: An open-source framework for federated learning. arXiv preprint arXiv:2105.06413, 2021.

[34] N. Rieke, J. Hancox, W. Li, F. Milletari, H. R. Roth, S. Albarqouni, S. Bakas, M. N. Galtier, B. A. Landman, K. Maier-Hein, et al. The future of digital health with federated learning. NPJ digital medicine, 3(1):1–7, 2020.

[35] H. R. Roth, K. Chang, P. Singh, N. Neumark, W. Li, V. Gupta, S. Gupta, L. Qu, A. Ihsani, B. C. Bizzo, et al. Federated learning for breast density classification: A real-world implementation. In Domain Adaptation and Representation Transfer, and Distributed and Collaborative Learning, pages 181–191. Springer, 2020.

[36] D. Rothchild, A. Panda, E. Ullah, N. Ivkin, I. Stoica, V. Braverman, J. Gonzalez, and R. Arora. Fetchsgd: Communication-efficient federated learning with sketching. In International Conference on Machine Learning, pages 8253–8265. PMLR, 2020.

[37] K. V. Sarma, S. Harmon, T. Sanford, H. R. Roth, Z. Xu, J. Tetreault, D. Xu, M. G. Flores, A. G. Raman, R. Kulkarni, et al. Federated learning improves site performance in multicenter deep learning without data sharing. Journal of the American Medical Informatics Association, 28(6):1259–1264, 2021.

[38] M. J. Sheller, B. Edwards, G. A. Reina, J. Martin, S. Pati, A. Kotrotsou, M. Milchenko, W. Xu, D. Marcus, R. R. Colen, et al. Federated learning in medicine: facilitating multi-institutional collaborations without sharing patient data. Scientific reports, 10(1):1–12, 2020.

[39] M. J. Sheller, G. A. Reina, B. Edwards, J. Martin, and S. Bakas. Multi-institutional deep learning modeling without sharing patient data: A feasibility study on brain tumor segmentation. In International MICCAI Brainlesion Workshop, pages 92–104. Springer, 2018.

[40] S. Silva, A. Altmann, B. Gutman, and M. Lorenzi. Fed-biomed: A general open-source frontend framework for federated learning in healthcare. In Domain Adaptation and Representation Transfer, and Distributed and Collaborative Learning, pages 201–210. Springer, 2020.

[41] H. Wang, M. Yurochkin, Y. Sun, D. Papailiopoulos, and Y. Khazaeni. Federated learning with matched averaging. arXiv preprint arXiv:2002.06440, 2020.

[42] S. Warnat-Herresthal, H. Schultze, K. L. Shastry, S. Manamohan, S. Mukherjee, V. Garg, R. Sarveswara, K. Händler, P. Pickkers, N. A. Aziz, et al. Swarm learning for decentralized and confidential clinical machine learning. Nature, 594(7862):265–270, 2021.

[43] Wikipedia contributors. Private set intersection — Wikipedia, the free encyclopedia, 2023. [Online; accessed 27-April-2023].

[44] Y. Xie, Z. Wang, D. Chen, D. Gao, L. Yao, W. Kuang, Y. Li, B. Ding, and J. Zhou. Federatedscope: A comprehensive and flexible federated learning platform via message passing. arXiv preprint arXiv:2204.05011, 2022.

[45] A. Xu, W. Li, P. Guo, D. Yang, H. R. Roth, A. Hatamizadeh, C. Zhao, D. Xu, H. Huang, and Z. Xu. Closing the generalization gap of cross-silo federated medical image segmentation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 20866–20875, 2022.

[46] Q. Yang, Y. Liu, T. Chen, and Y. Tong. Federated machine learning: Concept and applications. ACM Transactions on Intelligent Systems and Technology (TIST), 10(2):1–19, 2019.

[47] H. Zhao, W. Du, F. Li, P. Li, and G. Liu. Reduce communication costs and preserve privacy: Prompt tuning method in federated learning. arXiv preprint arXiv:2208.12268, 2022.

[48] A. Ziller, A. Trask, A. Lopardo, B. Szymkow, B. Wagner, E. Bluemke, J.-M. Nounahon, J. Passerat-Palmbach, K. Prakash, N. Rose, et al. Pysyft: A library for easy federated learning. In Federated Learning Systems, pages 111–139. Springer, 2021.

Data Engineering

It's FREE to join!

TCDE

tab.computer.org/tcde/

The Technical Committee on Data Engineering (TCDE) of the IEEE Computer Society is concerned with the role of data in the design, development, management and utilization of information systems.

- Data Management Systems and Modern Hardware/Software Platforms
- Data Models, Data Integration, Semantics and Data Quality
- Spatial, Temporal, Graph, Scientific, Statistical and Multimedia Databases
- Data Mining, Data Warehousing, and OLAP
- Big Data, Streams and Clouds
- Information Management, Distribution, Mobility, and the WWW
- Data Security, Privacy and Trust
- Performance, Experiments, and Analysis of Data Systems

The TCDE sponsors the International Conference on Data Engineering (ICDE). It publishes a quarterly newsletter, the Data Engineering Bulletin. If you are a member of the IEEE Computer Society, you may join the TCDE and receive copies of the Data Engineering Bulletin without cost. There are approximately 1000 members of the TCDE.

Join TCDE via Online or Fax

ONLINE: Follow the instructions on this page:

www.computer.org/portal/web/tandc/joinatc

FAX: Complete your details and fax this form to **+61-7-3365 3248**

Name _____

IEEE Member # _____

Mailing Address _____

_____

Country _____

Email _____

Phone _____

TCDE Mailing List

TCDE will occasionally email announcements, and other opportunities available for members. This mailing list will be used only for this purpose.

Membership Questions?

Xiaoyong Du
Key Laboratory of Data Engineering and Knowledge Engineering
Renmin University of China
Beijing 100872, China
duyong@ruc.edu.cn

TCDE Chair

Xiaofang Zhou
School of Information Technology and Electrical Engineering
The University of Queensland
Brisbane, QLD 4072, Australia
zxf@uq.edu.au

IEEE Computer Society
10662 Los Vaqueros Circle
Los Alamitos, CA 90720-1314

Non-profit Org.
U.S. Postage
PAID
Los Alamitos, CA
Permit 1398