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

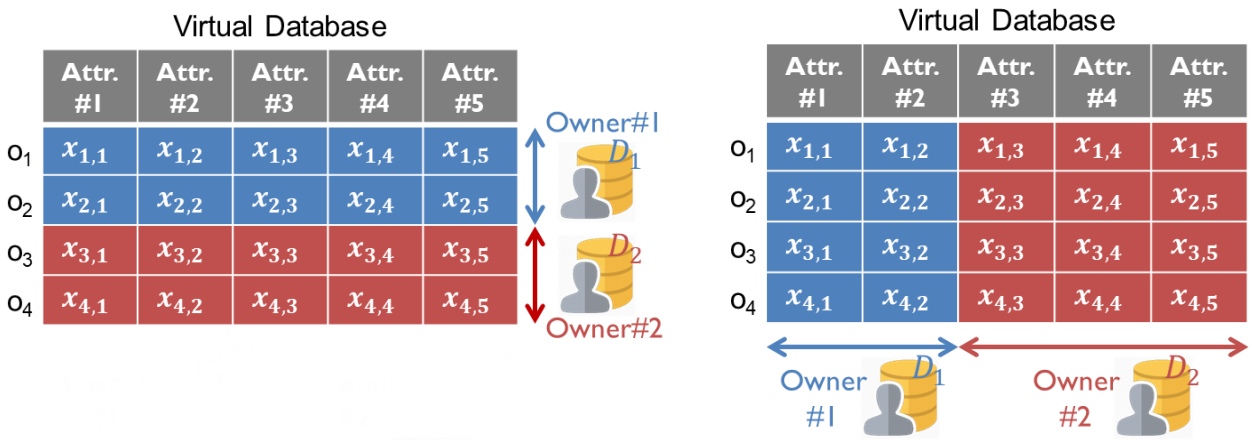
In the following, we introduce the key concepts in federated computing and the problem statement.

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).

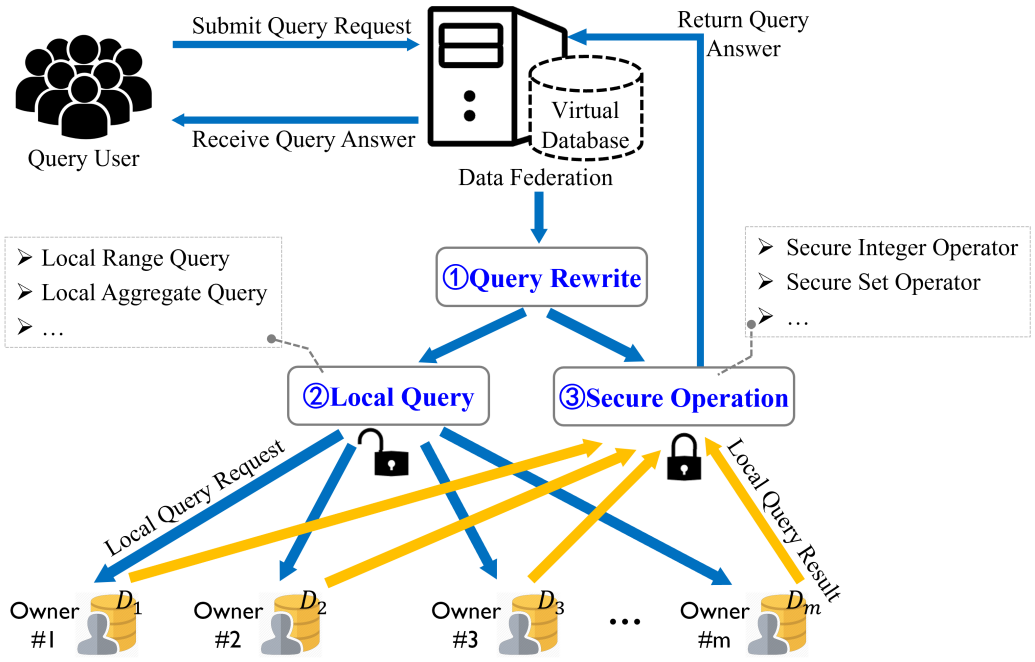


Figure 2: Workflow of processing federated queries over a data federation.

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.

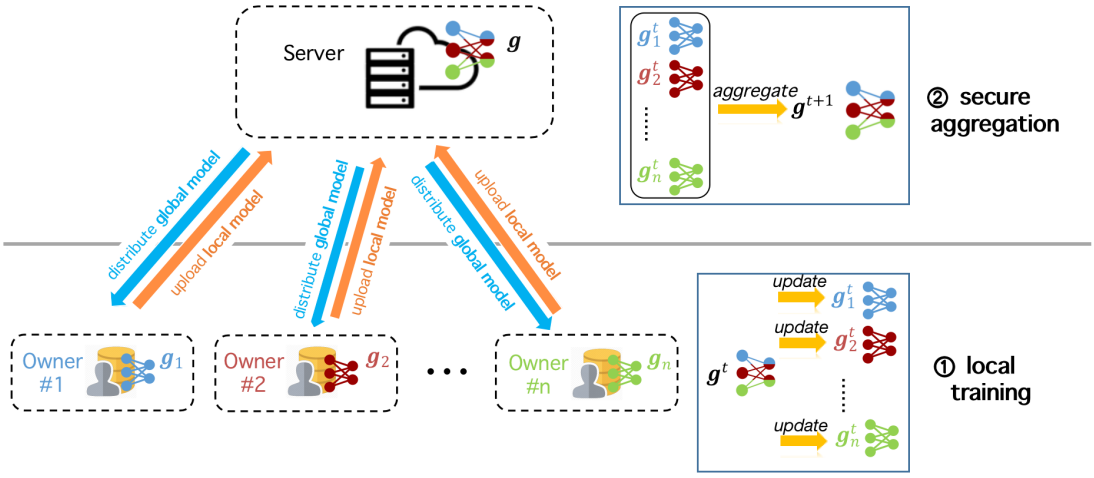


Figure 3: Workflow of federated learning over a data federation.

- **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.

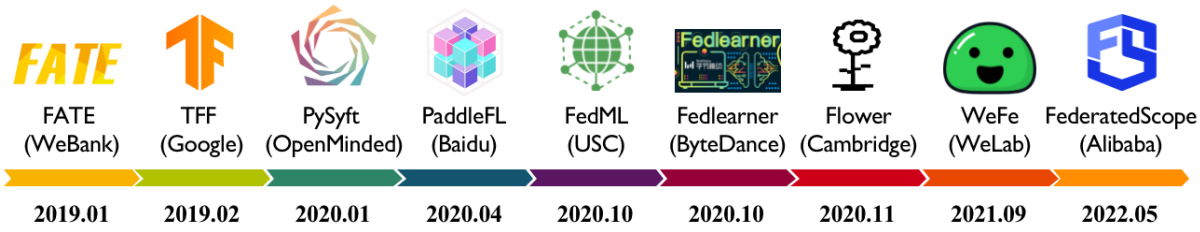


Figure 4: Representative frameworks for federated learning.

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.