Foundations and Trends® in Machine Learning

Advances and Open Problems in Federated Learning

Suggested Citation: Peter Kairouz, H. Brendan McMahan, *et al.* (2021), "Advances and Open Problems in Federated Learning", Foundations and Trends® in Machine Learning: Vol. 14, No. 1–2, pp 1–210. DOI: 10.1561/2200000083.

Peter Kairouz
Google Research
Kairouz@google.com

H. Brendan McMahan
Google Research

et al.

now
the essence of knowledge
Boston — Delft

Contents

Advances and Open Problems in Federated Learning

Peter Kairouz*[1], H. Brendan McMahan*[2] *et al.*

[1] *Google Research, USA; Kairouz@google.com*
[2] *Google Research, USA*

ABSTRACT

Federated learning (FL) is a machine learning setting where many clients (e.g., mobile devices or whole organizations) collaboratively train a model under the orchestration of a central server (e.g., service provider), while keeping the training data decentralized. FL embodies the principles of focused data collection and minimization, and can mitigate many of the systemic privacy risks and costs resulting from traditional, centralized machine learning and data science approaches. Motivated by the explosive growth in FL research, this monograph discusses recent advances and presents an extensive collection of open problems and challenges.

Peter Kairouz and H. Brendan McMahan conceived, coordinated, and edited this work.

Peter Kairouz, H. Brendan McMahan, *et al.* (2021), "Advances and Open Problems in Federated Learning", Foundations and Trends® in Machine Learning: Vol. 14, No. 1–2, pp 1–210. DOI: 10.1561/2200000083.

Full Author List

Peter Kairouz
Google Research

H. Brendan McMahan
Google Research

Brendan Avent
USC

Aurélien Bellet
INRIA

Mehdi Bennis
University of Oulu

Arjun Nitin Bhagoji
Princeton University

Kallista Bonawitz
Google Research

Zachary Charles
Google Research

Graham Cormode
University of Warwick

Rachel Cummings
Georgia Tech.

Rafael G. L. D'Oliveira
Rutgers University

Hubert Eichner
Google Research

Salim El Rouayheb
Rutgers University

David Evans
University of Virginia

Josh Gardner
University of Washington

Zachary Garrett
Google Research

Adrià Gascón
Google Research

Badih Ghazi
Google Research

Phillip B. Gibbons
CMU

Marco Gruteser
Google Research
Rutgers University

Zaid Harchaoui
University of Washington

Chaoyang He
USC

Lie He
EPFL

Zhouyuan Huo
University of Pittsburgh

Ben Hutchinson
Google Research

Justin Hsu
UW–Madison

Martin Jaggi
EPFL

Tara Javidi
UC San Diego

Gauri Joshi
CMU

Mikhail Khodak
CMU

Jakub Konecný
Google Research

Aleksandra Korolova
USC

Farinaz Koushanfar
UC San Diego

Sanmi Koyejo
Google Research
UIUC

Tancrède Lepoint
Google Research

Yang Liu
NTU

Prateek Mittal
Princeton

Mehryar Mohri
Google Research

Richard Nock
ANU

Ayfer Özgür
Stanford

Rasmus Pagh
Google Research
IT University of Copenhagen

Hang Qi
Google Research

Daniel Ramage
Google Research

Ramesh Raskar
MIT

Mariana Raykova
Google Research

Dawn Song
UC Berkeley

Weikang Song
Google Research

Sebastian U. Stich
EPFL

Ziteng Sun
Cornell

Ananda Theertha Suresh
Google Research

Florian Tramèr
Stanford

Praneeth Vepakomma
MIT

Jianyu Wang
CMU

Li Xiong
Emory

Zheng Xu
Google Research

Qiang Yang
HKUST

Felix X. Yu
Google Research

Han Yu
NTU

Sen Zhao
Google Research

1 Introduction

Federated learning (FL) is a machine learning setting where many clients (e.g., mobile devices or whole organizations) collaboratively train a model under the orchestration of a central server (e.g., service provider), while keeping the training data decentralized. It embodies the principles of focused collection and data minimization, and can mitigate many of the systemic privacy risks and costs resulting from traditional, centralized machine learning. This area has received significant interest recently, both from research and applied perspectives. This monograph describes the defining characteristics and challenges of the federated learning setting, highlights important practical constraints and considerations, and then enumerates a range of valuable research directions. The goals of this work are to highlight research problems that are of significant theoretical and practical interest, and to encourage research on problems that could have significant real-world impact.

The term *federated learning* was introduced in 2016 by McMahan *et al.* [1]: “We term our approach Federated Learning, since the learning task is solved by a loose federation of participating devices (which we refer to as clients) which are coordinated by a central server.” An unbalanced and non-IID (identically and independently distributed) data partitioning across a massive number of unreliable devices with limited communication bandwidth was introduced as the defining set of challenges.

Significant related work predates the introduction of the term federated learning. A longstanding goal pursued by many research communities (including cryptography, databases, and machine learning) is to analyze and learn from data distributed among many owners without exposing that data. Cryptographic methods for computing on encrypted data were developed starting in the early 1980s [2], [3], and Agrawal and Srikant [4] and Vaidya *et al.* [5] are early examples of work that sought to learn from local data using a centralized server while preserving privacy. Conversely, even since the introduction of the term federated learning, we are aware of no single work that directly addresses the full set of FL challenges. Thus, the term federated learning provides a convenient shorthand for a set of characteristics, constraints, and challenges that often co-occur in applied ML problems on decentralized data where privacy is paramount.

This monograph originated at the Workshop on Federated Learning and Analytics held June 17–18th, 2019, hosted at Google's Seattle office. During the course of this two-day event, the need for a broad paper surveying the many open challenges in the area of federated learning became clear.[1]

A key property of many of the problems discussed is that they are inherently interdisciplinary—solving them likely requires not just machine learning, but techniques from distributed optimization, cryptography, security, differential privacy, fairness, compressed sensing, systems, information theory, statistics, and more. Many of the hardest problems are at the intersections of these areas, and so we believe collaboration will be essential to ongoing progress. One of the goals of this work is to highlight the ways in which techniques from these fields can potentially be combined, raising both interesting possibilities as well as new challenges.

[1]During the preparation of this work, Li *et al.* [6] independently released an excellent but less comprehensive survey.

Since the term federated learning was initially introduced with an emphasis on mobile and edge device applications [1], [7], interest in applying FL to other applications has greatly increased, including some which might involve only a small number of relatively reliable clients, for example multiple organizations collaborating to train a model. We term these two federated learning settings "cross-device" and "cross-silo" respectively. Given these variations, we propose a somewhat broader definition of federated learning:

> **Federated learning** is a machine learning setting where multiple entities (clients) collaborate in solving a machine learning problem, under the coordination of a central server or service provider. Each client's raw data is stored locally and not exchanged or transferred; instead, focused updates intended for immediate aggregation are used to achieve the learning objective.

Focused updates are updates narrowly scoped to contain the minimum information necessary for the specific learning task at hand; aggregation is performed as early as possible in the service of data minimization. We note that this definition distinguishes federated learning from fully decentralized (peer-to-peer) learning techniques as discussed in Subsection 2.1.

Although privacy-preserving data analysis has been studied for more than 50 years, only in the past decade have solutions been widely deployed at scale (e.g., [8], [9]). Cross-device federated learning and federated data analysis are now being applied in consumer digital products. Google makes extensive use of federated learning in the Gboard mobile keyboard [10]–[14], as well as in features on Pixel phones and in Android Messages [15]. While Google has pioneered cross-device FL, interest in this setting is now much broader, for example: Apple is using cross-device FL in iOS 13 [16], for applications like the QuickType keyboard and the vocal classifier for "Hey Siri" [17]; doc.ai is developing cross-device FL solutions for medical research [18], and Snips has explored cross-device FL for hotword detection [19].

Cross-silo applications have also been proposed or described in myriad domains including finance risk prediction for reinsurance [20], pharmaceuticals discovery [21], electronic health records mining [22], medical data segmentation [23], [24], and smart manufacturing [25].

The growing demand for federated learning technology has resulted in a number of tools and frameworks becoming available. These include TensorFlow Federated [26], Federated AI Technology Enabler [27], PySyft [28], Leaf [29], PaddleFL [30] and Clara Training Framework [31]; more details in Appendix A.1. Commercial data platforms incorporating federated learning are in development from established technology companies as well as smaller start-ups.

Table 1.1 contrasts both cross-device and cross-silo federated learning with traditional single-datacenter distributed learning across a range of axes. These characteristics establish many of the constraints that practical federated learning systems must typically satisfy, and hence serve to both motivate and inform the open challenges in federated learning. They will be discussed at length in the sections that follow.

These two FL variants are called out as representative and important examples, but different FL settings may have different combinations of these characteristics. For the remainder of this monograph, we consider the cross-device FL setting unless otherwise noted, though many of the problems apply to other FL settings as well. Section 2 specifically addresses some of the many other variations and applications.

Next, we consider cross-device federated learning in more detail, focusing on practical aspects common to a typical large-scale deployment of the technology; Bonawitz *et al.* [32] provides even more detail for a particular production system, including a discussion of specific architectural choices and considerations.

1.1 The Cross-Device Federated Learning Setting

This section takes an applied perspective, and unlike the previous section, does not attempt to be definitional. Rather, the goal is to describe some of the practical issues in cross-device FL and how they might fit into a broader machine learning development and deployment ecosystem. The hope is to provide useful context and motivation for the open problems that follow, as well as to aid researchers in estimating how straightforward it would be to deploy a particular new approach

Table 1.1: Typical characteristics of federated learning settings vs. distributed learning in the datacenter (e.g., [33]). Cross-device and cross-silo federated learning are two examples of FL domains, but are not intended to be exhaustive. The primary defining characteristics of FL are highlighted in bold, but the other characteristics are also critical in determining which techniques are applicable

	Datacenter Distributed Learning	**Cross-Silo Federated Learning**	**Cross-Device Federated Learning**
Setting	Training a model on a large but “flat” dataset. Clients are compute nodes in a single cluster or datacenter.	Training a model on siloed data. Clients are different organizations (e.g., medical or financial) or geo-distributed datacenters.	The clients are a very large number of mobile or IoT devices.
Data distribution	Data is centrally stored and can be shuffled and balanced across clients. Any client can read any part of the dataset.	**Data is generated locally and remains decentralized.** Each client stores its own data and cannot read the data of other clients. Data is not independently or identically distributed.	
Orchestration	Centrally orchestrated.	**A central orchestration server/service organizes the training**, but never sees raw data.	
Wide-area communication	None (fully connected clients in one datacenter/cluster).	Typically a hub-and-spoke topology, with the hub representing a coordinating service provider (typically without data) and the spokes connecting to clients.	
Data availability	——— All clients are almost always available.———		Only a fraction of clients are available at any one time, often with diurnal or other variations.
Distribution scale	Typically 1–1000 clients.	Typically 2–100 clients.	Massively parallel, up to 10^{10} clients.
Primary bottleneck	Computation is more often the bottleneck in the datacenter, where very fast networks can be assumed.	Might be computation or communication.	Communication is often the primary bottleneck, though it depends on the task. Generally, cross-device federated computations use wi-fi or slower connections.

Continued.

Table 1.1: Continued

	Datacenter Distributed Learning	**Cross-Silo Federated Learning**	**Cross-Device Federated Learning**
Addressability	Each client has an identity or name that allows the system to access it specifically.		Clients cannot be indexed directly (i.e., no use of client identifiers).
Client statefulness	Stateful—each client may participate in each round of the computation, carrying state from round to round.		Stateless—each client will likely participate only once in a task, so generally a fresh sample of never-before-seen clients in each round of computation is assumed.
Client reliability	Relatively few failures.		Highly unreliable—5% or more of the clients participating in a round of computation are expected to fail or drop out (e.g., because the device becomes ineligible when battery, network, or idleness requirements are violated).
Data partition axis	Data can be partitioned/re-partitioned arbitrarily across clients.	Partition is fixed. Could be example-partitioned (horizontal) or feature-partitioned (vertical).	Fixed partitioning by example (horizontal).

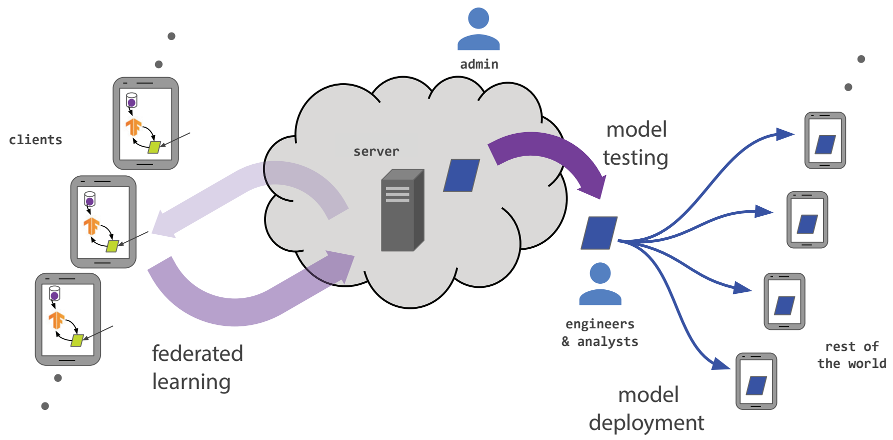


Figure 1.1: The lifecycle of an FL-trained model and the various actors in a federated learning system. This figure is revisited in Section 4 from a threat models perspective.

in a real-world system. We begin by sketching the lifecycle of a model before considering a FL training process.

1.1.1 The Lifecycle of a Model in Federated Learning

The FL process is typically driven by a model engineer developing a model for a particular application. For example, a domain expert in natural language processing may develop a next word prediction model for use in a virtual keyboard. Figure 1.1 shows the primary components and actors. At a high level, a typical workflow is:

1. **Problem identification:** The model engineer identifies a problem to be solved with FL.

2. **Client instrumentation:** If needed, the clients (e.g., an app running on mobile phones) are instrumented to store locally (with limits on time and quantity) the necessary training data. In many cases, the app already will have stored this data (e.g., a text messaging app must store text messages, a photo management app already stores photos). However, in some cases additional data or metadata might need to be maintained, e.g., user interaction data to provide labels for a supervised learning task.

3. **Simulation prototyping (optional):** The model engineer may prototype model architectures and test learning hyperparameters in an FL simulation using a proxy dataset.

4. **Federated model training:** Multiple federated training tasks are started to train different variations of the model, or use different optimization hyperparameters.

5. **(Federated) model evaluation:** After the tasks have trained sufficiently (typically a few days, see below), the models are analyzed and good candidates selected. Analysis may include metrics computed on standard datasets in the datacenter, or federated evaluation wherein the models are pushed to held-out clients for evaluation on local client data.

6. **Deployment:** Finally, once a good model is selected, it goes through a standard model launch process, including manual quality assurance, live A/B testing (usually by using the new model on some devices and the previous generation model on other devices to compare their in-vivo performance), and a staged rollout (so that poor behavior can be discovered and rolled back before affecting too many users). The specific launch process for a model is set by the owner of the application and is usually independent of how the model is trained. In other words, this step would apply equally to a model trained with federated learning or with a traditional datacenter approach.

One of the primary practical challenges an FL system faces is making the above workflow as straightforward as possible, ideally approaching the ease-of-use achieved by ML systems for centralized training. While much of this monograph concerns federated training specifically, there are many other components including federated analytics tasks like model evaluation and debugging. Improving these is the focus of Subsection 3.4. For now, we consider in more detail the training of a single FL model (Step 4 above).

1.1.2 A Typical Federated Training Process

We now consider a template for FL training that encompasses the Federated Averaging algorithm of McMahan *et al.* [1] and many others; again, variations are possible, but this gives a common starting point.

A server (service provider) orchestrates the training process, by repeating the following steps until training is stopped (at the discretion of the model engineer who is monitoring the training process):

1. **Client selection:** The server samples from a set of clients meeting eligibility requirements. For example, mobile phones might only check in to the server if they are plugged in, on an unmetered wi-fi connection, and idle, in order to avoid impacting the user of the device.

2. **Broadcast:** The selected clients download the current model weights and a training program (e.g., a TensorFlow graph [34]) from the server.

3. **Client computation:** Each selected device locally computes an update to the model by executing the training program, which might for example run SGD on the local data (as in Federated Averaging).

4. **Aggregation:** The server collects an aggregate of the device updates. For efficiency, stragglers might be dropped at this point once a sufficient number of devices have reported results. This stage is also the integration point for many other techniques which will be discussed later, possibly including: secure aggregation for added privacy, lossy compression of aggregates for communication efficiency, and noise addition and update clipping for differential privacy.

5. **Model update:** The server locally updates the shared model based on the aggregated update computed from the clients that participated in the current round.

Table 1.2 gives typical order-of-magnitude sizes for the quantities involved in a typical federated learning application on mobile devices.

Table 1.2: Order-of-magnitude sizes for typical cross-device federated learning applications

Total population size	10^6–10^{10} devices
Devices selected for one round of training	50–5000
Total devices that participate in training one model	10^5–10^7
Number of rounds for model convergence	500–10000
Wall-clock training time	1–10 days

The separation of the client computation, aggregation, and model update phases is not a strict requirement of federated learning, and it indeed excludes certain classes of algorithms, for example asynchronous SGD where each client's update is immediately applied to the model, before any aggregation with updates from other clients. Such asynchronous approaches may simplify some aspects of system design, and also be beneficial from an optimization perspective (though this point can be debated). However, the approach presented above has a substantial advantage in affording a separation of concerns between different lines of research: advances in compression, differential privacy, and secure multi-party computation can be developed for standard primitives like computing sums or means over decentralized updates, and then composed with arbitrary optimization or analytics algorithms, so long as those algorithms are expressed in terms of aggregation primitives.

It is also worth emphasizing that in two respects, the FL training process should not impact the user experience. First, as outlined above, even though model parameters are typically sent to some devices during the broadcast phase of each round of federated training, these models are an ephemeral part of the training process, and not used to make "live" predictions shown to the user. This is crucial, because training ML models is challenging, and a misconfiguration of hyperparameters can produce a model that makes bad predictions. Instead, user-visible use of the model is deferred to a rollout process as detailed above in Step 6 of the model lifecycle. Second, the training itself is intended to be invisible to the user—as described under client selection, training does not slow the device or drain the battery because it only executes when the device is idle and connected to power. However, the limited availability these constraints introduce leads directly to open research challenges which will be discussed subsequently, such as semi-cyclic data availability and the potential for bias in client selection.

1.2 Federated Learning Research

The remainder of this monograph surveys many open problems that are motivated by the constraints and challenges of real-world federated learning settings, from training models on medical data from a hospital system to training using hundreds of millions of mobile devices. Needless to say, most researchers working on federated learning problems will likely not be deploying production FL systems, nor have access to fleets of millions of real-world devices. This leads to a key distinction between the practical settings that motivate the work and experiments conducted in simulation which provide evidence of the suitability of a given approach to the motivating problem.

This makes FL research somewhat different than other ML fields from an experimental perspective, leading to additional considerations in conducting FL research. In particular, when highlighting open problems, we have attempted, when possible, to also indicate relevant performance metrics which can be measured in simulation, the characteristics of datasets which will make them more representative of real-world performance, etc. The need for simulation also has ramifications for the presentation of FL research. While not intended to be authoritative or absolute, we make the following modest suggestions for presenting FL research that addresses the open problems we describe:

- As shown in Table 1.1, the FL setting can encompass a wide range of problems. Compared to fields where the setting and goals are well-established, it is important to precisely describe the details of the particular FL setting of interest, particularly when the proposed approach makes assumptions that may not be appropriate in all settings (e.g., stateful clients that participate in all rounds).
- Of course, details of any simulations should be presented in order to make the research reproducible. But it is also important to

explain which aspects of the real-world setting the simulation is designed to capture (and which it is not), in order to effectively make the case that success on the simulated problem implies useful progress on the real-world objective. We hope that the guidance in this monograph will help with this.

- Privacy and communication efficiency are always first-order concerns in FL, even if the experiments are simulations running on a single machine using public data. More so than with other types of ML, for any proposed approach it is important to be unambiguous about *where computation happens* as well as *what is communicated.*

Software libraries for federated learning simulation as well as standard datasets can help ease the challenges of conducting effective FL research; Appendix A.1 summarizes some of the currently available options. Developing standard evaluation metrics and establishing standard benchmark datasets for different federated learning settings (cross-device and cross-silo) remain highly important directions for ongoing work.

1.3 Organization

Section 2 builds on the ideas in Table 1.1, exploring other FL settings and problems beyond the original focus on cross-device settings. Section 3 then turns to core questions around improving the efficiency and effectiveness of federated learning. Section 4 undertakes a careful consideration of threat models and considers a range of technologies toward the goal of achieving rigorous privacy protections. As with all machine learning systems, in federated learning applications there may be incentives to manipulate the models being trained, and failures of various kinds are inevitable; these challenges are discussed in Section 5. Finally, we address the important challenges of providing fair and unbiased models in Section 6.

2

Relaxing the Core FL Assumptions: Applications to Emerging Settings and Scenarios

In this section, we will discuss areas of research related to the topics discussed in the previous section. Even though not being the main focus of the remainder of the monograph, progress in these areas could motivate design of the next generation of production systems.

2.1 Fully Decentralized/Peer-to-Peer Distributed Learning

In federated learning, a central server orchestrates the training process and receives the contributions of all clients. The server is thus a central player which also potentially represents a single point of failure. While large companies or organizations can play this role in some application scenarios, a reliable and powerful central server may not always be available or desirable in more collaborative learning scenarios [35]. Furthermore, the server may even become a bottleneck when the number of clients is very large, as demonstrated by Lian *et al.* [36] (though this can be mitigated by careful system design, e.g., [32]).

The key idea of fully decentralized learning is to replace communication with the server by peer-to-peer communication between individual clients. The communication topology is represented as a connected graph in which nodes are the clients and an edge indicates a communication channel between two clients. The network graph is typically chosen to be sparse with small maximum degree so that each node only needs to send/receive messages to/from a small number of peers; this is in contrast to the star graph of the server-client architecture. In fully decentralized algorithms, a round corresponds to each client performing a local update and exchanging information with their neighbors in the graph.[1] In the context of machine learning, the local update is typically a local (stochastic) gradient step and the communication consists in averaging one's local model parameters with the neighbors. Note that there is no longer a global state of the model as in standard federated learning, but the process can be designed such that all local models converge to the desired global solution, i.e., the individual models gradually reach consensus. While multi-agent optimization has a long history in the control community, fully decentralized variants of SGD and other optimization algorithms have recently been considered in machine learning both for improved scalability in datacenters [38] as well as for decentralized networks of devices [35], [39]–[44]. They consider undirected network graphs, although the case of directed networks (encoding unidirectional channels which may arise in real-world scenarios such as social networks or data markets) has also been studied in [38], [45].

It is worth noting that even in the decentralized setting outlined above, a central authority may still be in charge of setting up the learning task. Consider for instance the following questions: Who decides what is the model to be trained in the decentralized setting? What algorithm to use? What hyperparameters? Who is responsible for debugging when something does not work as expected? A certain degree of trust of the participating clients in a central authority would still be needed to answer these questions. Alternatively, the decisions could be taken by the client who proposes the learning task, or collaboratively through a consensus scheme (see Subsection 2.1.2).

[1]Note, however, that the notion of a round does not need to even make sense in this setting. See for instance the discussion on clock models in [37].

Table 2.1: A comparison of the key distinctions between federated learning and fully decentralized learning. Note that as with FL, decentralized learning can be further divided into different use-cases, with distinctions similar to those made in Table 1.1 comparing cross-silo and cross-device FL

	Federated Learning	**Fully Decentralized (Peer-to-Peer) Learning**
Orchestration	A central orchestration server or service organizes the training, but never sees raw data.	No centralized orchestration.
Wide-area communication	Typically a hub-and-spoke topology, with the hub representing a coordinating service provider (typically without data) and the spokes connecting to clients.	Peer-to-peer topology, with a possibly dynamic connectivity graph.

Table 2.1 provides a comparison between federated and peer-to-peer learning. While the architectural assumptions of decentralized learning are distinct from those of federated learning, it can often be applied to similar problem domains, many of the same challenges arise, and there is significant overlap in the research communities. Thus, we consider decentralized learning in this monograph as well; in this section challenges specific to the decentralized approach are explicitly considered, but many of the open problems in other sections also arise in the decentralized case.

2.1.1 Algorithmic Challenges

A large number of important algorithmic questions remain open on the topic of real-world usability of decentralized schemes for machine learning. Some questions are analogous to the special case of federated learning with a central server, and other challenges come as an additional side-effect of being fully decentralized or trust-less. We outline some particular areas in the following.

Effect of Network Topology and Asynchrony on Decentralized SGD Fully decentralized algorithms for learning should be robust to the limited availability of the clients (with clients temporarily unavailable, dropping out or joining during the execution) and limited reliability of the network (with possible message drops). While for the special case of generalized linear models, schemes using the duality structure could enable some of these desired robustness properties [46], for the case of deep learning and SGD this remains an open question. When the network graph is complete but messages have a fixed probability to be dropped, Yu *et al.* [47] show that one can achieve convergence rates that are comparable to the case of a reliable network. Additional open research questions concern non-IID data distributions, update frequencies, efficient communication patterns and practical convergence time [44], as we outline in more detail below.

Well-connected or denser networks encourage faster consensus and give better theoretical convergence rates, which depend on the spectral gap of the network graph. However, when data is IID, sparser topologies do not necessarily hurt the convergence in practice: this was analyzed theoretically in [48]. Denser networks typically incur communication delays which increase with the node degrees. Most of optimization-theory works do not explicitly consider how the topology affects the runtime, that is, wall-clock time required to complete each SGD iteration. Wang *et al.* [49] propose MATCHA, a decentralized SGD method based on matching decomposition sampling, that reduces the communication delay per iteration for any given node topology while maintaining the same error convergence speed. The key idea is to decompose the graph topology into matchings consisting of disjoint communication links that can operate in parallel, and carefully choose a subset of these matchings in each iteration. This sequence of subgraphs results in more frequent communication over connectivity-critical links (ensuring fast error convergence) and less frequent communication over other links (saving communication delays).

The setting of decentralized SGD also naturally lends itself to asynchronous algorithms in which each client becomes active independently at random times, removing the need for global synchronization and potentially improving scalability [35], [38]–[40], [50].

Local-Update Decentralized SGD The theoretical analysis of schemes which perform several local update steps before a communication round is significantly more challenging than those using a single SGD step, as in mini-batch SGD. While this will also be discussed later in Subsection 3.2, the same also holds more generally in the fully decentralized setting of interest here. Schemes relying on a single local update step are typically proven to converge in the case of non-IID local datasets [42], [51]. For the case with several local update steps, [52], [53] recently provided convergence analysis. Further, [49] provides a convergence analysis for the non-IID data case, but for the specific scheme based on matching decomposition sampling described above. In general, however, understanding the convergence under non-IID data distributions and how to design a model averaging policy that achieves the fastest convergence remains an open problem.

Personalization, and Trust Mechanisms Similarly to the cross-device FL setting, an important task for the fully decentralized scenario under the non-IID data distributions available to individual clients is to design algorithms for learning collections of personalized models. The work of [35], [39] introduces fully decentralized algorithms to collaboratively learn a personalized model for each client by smoothing model parameters across clients that have similar tasks (i.e., similar data distributions). Zantedeschi *et al.* [54] further learn the similarity graph together with the personalized models. One of the key unique challenges in the decentralized setting remains the robustness of such schemes to malicious actors or contribution of unreliable data or labels. The use of incentives or mechanism design in combination with decentralized learning is an emerging and important goal, which may be harder to achieve in the setting without a trusted central server.

Gradient Compression and Quantization Methods In potential applications, the clients would often be limited in terms of communication bandwidth available and energy usage permitted. Translating and generalizing some of the existing compressed communication schemes from the centralized orchestrator-facilitated setting (see Subsection 3.5) to the fully decentralized setting, without negatively impacting the convergence is an active research direction [42], [51], [55], [56]. A complementary idea is to design decentralized optimization algorithms which naturally give rise to sparse updates [54].

Privacy An important challenge in fully decentralized learning is to prevent any client from reconstructing the private data of another client from its shared updates while maintaining a good level of utility for the learned models. Differential privacy (see Section 4) is the standard approach to mitigate such privacy risks. In decentralized federated learning, this can be achieved by having each client add noise locally, as done in [39], [57]. Unfortunately, such local privacy approaches often come at a large cost in utility. Furthermore, distributed methods based on secure aggregation or secure shuffling that are designed to improve the privacy-utility trade-off in the standard FL setting (see Subsection 4.4.3) do not easily integrate with fully decentralized algorithms. A possible direction to achieve better trade-offs between privacy and utility in fully decentralized algorithms is to rely on decentralization itself to amplify differential privacy guarantees, for instance by considering appropriate relaxations of local differential privacy [58].

2.1.2 Practical Challenges

An orthogonal question for fully decentralized learning is how it can be practically realized. This section outlines a family of related ideas based on the idea of a distributed ledger, but other approaches remain unexplored.

A blockchain is a distributed ledger shared among disparate users, making possible digital transactions, including transactions of cryptocurrency, without a central authority. In particular, smart contracts allow execution of arbitrary code on top of the blockchain, essentially a massively replicated eventually-consistent state machine. In terms of federated learning, use of the technology could enable decentralization of the global server by using smart contracts to do model aggregation, where the participating clients executing the smart contracts could be different companies or cloud services.

However, on today's blockchain platforms such as Ethereum [59], data on the blockchains is publicly available by default, this could discourage users from participating in the decentralized federated learning protocol, as the protection of the data is typically the primary motivating factor for FL. To address such concerns, it might be possible to modify the existing privacy-preserving techniques to fit into the scenario of decentralized federated learning. First of all, to prevent the participating nodes from exploiting individually submitted model updates, existing secure aggregation protocols could be used. A practical secure aggregation protocol already used in cross-device FL was proposed by Bonawitz *et al.* [60], effectively handling dropping out participants at the cost of complexity of the protocol. An alternative system would be to have each client stake a deposit of cryptocurrency on blockchain, and get penalized if they drop out during the execution. Without the need of handling dropouts, the secure aggregation protocol could be significantly simplified. Another way of achieving secure aggregation is to use confidential smart contract such as what is enabled by the Oasis Protocol [61] which runs inside secure enclaves. With this, each client could simply submit an encrypted local model update, knowing that the model will be decrypted and aggregated inside the secure hardware through remote attestation (though see discussion of privacy-in-depth in Subsection 4.1).

In order to prevent any client from trying to reconstruct the private data of another client by exploiting the global model, client-level differential privacy [62] has been proposed for FL. Client-level differential privacy is achieved by adding random Gaussian noise on the aggregated global model that is enough to hide any single client's update. In the context of blockchain, each client could locally add a certain amount of Gaussian noise after local gradient descent steps and submit the model to blockchain. The local noise scale should be calculated such that the aggregated noise on blockchain is able to achieve the same client-level differential privacy as in [62]. Finally, the aggregated global model on blockchain could be encrypted and only the participating clients hold the decryption key, which protects the model from the public.

2.2 Cross-Silo Federated Learning

In contrast with the characteristics of cross-device federated learning, see Table 1.1, cross-silo federated learning admits more flexibility in certain aspects of the overall design, but at the same time presents a setting where achieving other properties can be harder. This section discusses some of these differences.

The cross-silo setting can be relevant where a number of companies or organizations share incentive to train a model based on all of their data, but cannot share their data directly. This could be due to constraints imposed by confidentiality or due to legal constraints, or even within a single company when they cannot centralize their data between different geographical regions. These cross-silo applications have attracted substantial attention.

Data Partitioning In the cross-device setting the data is assumed to be partitioned by examples. In the cross-silo setting, in addition to partitioning by examples, partitioning by features is of practical relevance. An example could be when two companies in different businesses have the same or overlapping set of customers, such as a local bank and a local retail company in the same city. This difference has been also referred to as horizontal and vertical federated learning by Yang *et al.* [63].

Cross-silo FL with data partitioned by features, employs a very different training architecture compared to the setting with data partitioned by example. It may or may not involve a central server as a neutral party, and based on specifics of the training algorithm, clients exchange specific intermediate results rather than model parameters, to assist other parties' gradient calculations; see for instance [63, Subsection 2.4.2]. In this setting, application of techniques such as secure multi-party computation or homomorphic encryption have been proposed in order to limit the amount of information other participants can infer from observing the training process. The downside of this approach is that the training algorithm is typically dependent on the type of machine learning objective being pursued. Currently proposed algorithms include trees [64], linear and logistic regression [63], [65], [66], and neural networks [67]. Local updates similar to Federated Averaging (see Subsection 3.2) has been proposed to address the communication challenges of feature-partitioned systems [66], and [68], [69] study the security and privacy related challenges inherent in such systems.

Federated transfer learning [63] is another concept that considers challenging scenarios in which data parties share only a partial overlap in the user space or the feature space, and leverage existing transfer learning techniques [70] to build models collaboratively. The existing formulation is limited to the case of two clients.

Partitioning by examples is usually relevant in cross-silo FL when a single company cannot centralize their data due to legal constraints, or when organizations with similar objectives want to collaboratively improve their models. For instance, different banks can collaboratively train classification or anomaly detection models for fraud detection [20], hospitals can build better diagnostic models [24], and so on.

An open-source platform supporting the above outlined applications is currently available as *Federated AI Technology Enabler (FATE)* [27]. At the same time, the IEEE P3652.1 Federated Machine Learning Working Group is focusing on standard-setting for the Federated AI Technology Framework. Other platforms include [31] focused on a range of medical applications and [71] for enterprise use cases. See Appendix A.1 for more details.

Incentive Mechanisms In addition to developing new algorithmic techniques for FL, incentive mechanism design for honest participation is an important practical research question. This need may arise in cross-device settings (e.g., [72], [73]), but is particularly relevant in the cross-silo setting, where participants may at the same time also be business competitors. The incentive can be in the form of monetary payout [74] or final models with different levels of performance [75]. The option to deliver models with performance commensurate to the contributions of each client is especially relevant in collaborative learning situations in which competitions exist among FL participants. Clients might worry that contributing their data to training federated learning models will benefit their competitors, who do not contribute as much but receive the same final model nonetheless (i.e., the free-rider problem). Related objectives include how to divide earnings generated by the federated learning model among contributing data owners in order to sustain long-term participation, and also how to link the incentives with decisions on defending against adversarial data owners to enhance system security, optimizing the participation of data owners to enhance system efficiency.

Differential Privacy The discussion of actors and threat models in Subsection 4.1 is largely relevant also for the cross-silo FL. However, protecting against different actors might have different priorities. For example, in many practical scenarios, the final trained model would be released only to those who participate in the training, which makes the concerns about "the rest of the world" less important.

On the other hand, for a practically persuasive claim, we would usually need a notion of local differential privacy, as the potential threat from other clients is likely to be more important. In cases when the clients are not considered a significant threat, each client could control the data from a number of their respective users, and a formal privacy guarantee might be needed on such user-level basis. Depending on application, other objectives could be worth pursuing. This area has not been systematically explored.

Tensor Factorization Several works have also studied cross-silo federated tensor factorization where multiple sites (each having a set of data with the same feature, i.e., horizontally partitioned) jointly perform tensor factorization by only sharing intermediate factors with the coordination server while keeping data private at each site. Among the existing works, [76] used an alternating direction method of multipliers (ADMM) based approach and [77] improved the efficiency with the elastic averaging SGD (EASGD) algorithm and further ensures differential privacy for the intermediate factors.

2.3 Split Learning

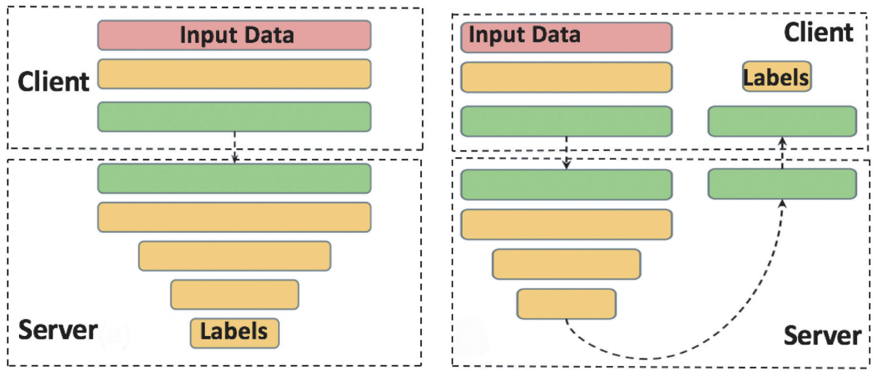
In contrast with the previous settings which focus on data partitioning and communication patterns, the key idea behind split learning [78], [79][2] is to split the execution of a model on a per-layer basis between the clients and the server. This can be done for both training and inference.

In the simplest configuration of split learning, each client computes the forward pass through a deep network up to a specific layer referred to as the *cut layer*. The outputs at the cut layer, referred to as *smashed data*, are sent to another entity (either the server or another client), which completes the rest of the computation. This completes a round of forward propagation without sharing the raw data. The gradients can then be back propagated from its last layer until the cut layer in a similar fashion. The gradients at the cut layer—and only these gradients—are sent back to the clients, where the rest of back propagation is completed. This process is continued until convergence, without having clients directly access each others raw data. This setup is shown in Figure 2.1(a) and a variant of this setup where labels are also not shared along with raw data is shown in Figure 2.1(b). Split learning approaches for data partitioned by features have been studied in [80].

In several settings, the overall communication requirements of split learning and federated learning were compared in [81]. Split learning brings in another dimension of parallelism in the training, parallelization among parts of a model, e.g., client and server. The ideas in [82], [83], where the authors break the dependencies between partial networks and reduced total centralized training time by parallelizing the computations in different parts, can be relevant here as well. However, it is still an open question to explore such parallelization of split learning on edge devices. Split learning also enables matching client-side model components with the best server-side model components for automating model selection as shown in the ExpertMatcher [84].

The values communicated can nevertheless, in general, reveal information about the underlying data. How much, and whether this

[2]See also split learning project website https://splitlearning.github.io/.



(a) Vanilla split learning (b) U-shaped split learning

Figure 2.1: Split learning configurations showing raw data is not transferred in the vanilla setting and that raw data as well as labels are not transferred between the client and server entities in the U-shaped split learning setting.

is acceptable, is likely going to be application and configuration specific. A variation of split learning called NoPeek SplitNN [85] reduces the potential leakage via communicated activations, by reducing their distance correlation [86], [87] with the raw data, while maintaining good model performance via categorical cross-entropy. The key idea is to minimize the distance correlation between the raw data points and communicated smashed data. The objects communicated could otherwise contain information highly correlated with the input data if used without NoPeek SplitNN, the use of which also enables the split to be made relatively early-on given the decorrelation it provides. One other engineering driven approach to minimize the amount of information communicated in split learning has been via a specifically learnt pruning of channels present in the client side activations [88]. Overall, much of the discussion in Section 4 is relevant here as well, and analysis providing formal privacy guarantees specifically for split learning is still an open problem.

2.4 Executive Summary

The motivation for federated learning is relevant for a number of related areas of research.

- Fully decentralized learning (Subsection 2.1) removes the need for a central server coordinating the overall computation. Apart from algorithmic challenges, open problems are in practical realization of the idea and in understanding of what form of trusted central authority is needed to set up the task.
- Cross-silo federated learning (Subsection 2.2) admits problems with different kinds of modelling constraints, such as data partitioned by examples and/or features, and faces different set of concerns when formulating formal privacy guarantees or incentive mechanisms for clients to participate.
- Split learning (Subsection 2.3) is an approach to partition the execution of a model between the clients and the server. It can deliver different options for overall communication constraints, but detailed analysis of when the communicated values reveal sensitive information is still missing.

3

Improving Efficiency and Effectiveness

In this section we explore a variety of techniques and open questions that address the challenge of making federated learning more efficient and effective. This encompasses a myriad of possible approaches, including: developing better optimization algorithms; providing different models to different clients; making ML tasks like hyperparameter search, architecture search, and debugging easier in the FL context; improving communication efficiency; and more.

One of the fundamental challenges in addressing these goals is the presence of non-IID data, so we begin by surveying this issue and highlighting potential mitigations.

3.1 Non-IID Data in Federated Learning

While the meaning of IID is generally clear, data can be non-IID in many ways. In this section, we provide a taxonomy of non-IID data regimes that may arise for any client-partitioned dataset. The most common sources of dependence and non-identicalness are due to each client corresponding to a particular user, a particular geographic location, and/or a particular time window. This taxonomy has a close mapping to notions of dataset shift [89], [90], which studies differences between the training distribution and testing distribution; here, we consider differences in the data distribution on each client.

For the following, consider a supervised task with features x and labels y. A statistical model of federated learning involves two levels of sampling: accessing a datapoint requires first sampling a client $i \sim \mathcal{Q}$, the distribution over available clients, and then drawing an example $(x, y) \sim \mathcal{P}_i(x, y)$ from that client's local data distribution.

When non-IID data in federated learning is referenced, this typically refers to differences between $\mathcal{P}_i$ and $\mathcal{P}_j$ for different clients i and j. However, it is also important to note that the distribution $\mathcal{Q}$ and $\mathcal{P}_i$ may change over time, introducing another dimension of "non-IIDness".

For completeness, we note that even considering the dataset on a single device, if the data is in an insufficiently-random order, e.g., ordered by time, then independence is violated locally as well. For example, consecutive frames in a video are highly correlated. Sources of intra-client correlation can generally be resolved by local shuffling.

Non-Identical Client Distributions We first survey some common ways in which data tend to deviate from being identically distributed, that is $P_i \neq P_j$ for different clients i and j. Rewriting $P_i(x, y)$ as $P_i(y \mid x)P_i(x)$ and $P_i(x \mid y)P_i(y)$ allows us to characterize the differences more precisely.

- *Feature distribution skew* (covariate shift): The marginal distributions $\mathcal{P}_i(x)$ may vary across clients, even if $\mathcal{P}(y \mid x)$ is shared.[1] For example, in a handwriting recognition domain, users who write the same words might still have different stroke width, slant, etc.

- *Label distribution skew* (prior probability shift): The marginal distributions $\mathcal{P}_i(y)$ may vary across clients, even if $\mathcal{P}(x \mid y)$ is the same. For example, when clients are tied to particular geo-regions, the distribution of labels varies across clients—kangaroos are only in Australia or zoos; a person's face is only in a few locations worldwide; for mobile device keyboards, certain emoji are used by one demographic but not others.

[1]We write "$\mathcal{P}(y \mid x)$ is shared" as shorthand for $\mathcal{P}_i(y \mid x) = \mathcal{P}_j(y \mid x)$ for all clients i and j.

- *Same label, different features* (concept drift): The conditional distributions $\mathcal{P}_i(x \mid y)$ may vary across clients even if $\mathcal{P}(y)$ is shared. The same label y can have very different features x for different clients, e.g., due to cultural differences, weather effects, standards of living, etc. For example, images of homes can vary dramatically around the world and items of clothing vary widely. Even within the U.S., images of parked cars in the winter will be snow-covered only in certain parts of the country. The same label can also look very different at different times, and at different time scales: day vs. night, seasonal effects, natural disasters, fashion and design trends, etc.

- *Same features, different label* (concept shift): The conditional distribution $\mathcal{P}_i(y \mid x)$ may vary across clients, even if $\mathcal{P}(x)$ is the same. Because of personal preferences, the same feature vectors in a training data item can have different labels. For example, labels that reflect sentiment or next word predictors have personal and regional variation.

- *Quantity skew* or unbalancedness: Different clients can hold vastly different amounts of data.

Real-world federated learning datasets likely contain a mixture of these effects, and the characterization of cross-client differences in real-world partitioned datasets is an important open question. Most empirical work on synthetic non-IID datasets (e.g., [1], [91]) have focused on label distribution skew, where a non-IID dataset is formed by partitioning a "flat" existing dataset based on the labels. A better understanding of the nature of real-world non-IID datasets will allow for the construction of controlled but realistic non-IID datasets for testing algorithms and assessing their resilience to different degrees of client heterogeneity.

Further, different non-IID regimes may require the development of different mitigation strategies. For example, under feature-distribution skew, because $\mathcal{P}(y \mid x)$ is assumed to be common, the problem is at least in principle well specified, and training a single global model that learns $\mathcal{P}(y \mid x)$ may be appropriate. When the same features map to different labels on different clients, some form of personalization (Subsection 3.3) may be essential to learning the true labeling functions.

Violations of Independence Violations of independence are introduced any time the distribution $\mathcal{Q}$ changes over the course of training; a prominent example is in cross-device FL, where devices typically need to meet eligibility requirements in order to participate in training (see Subsection 1.1.2). Devices typically meet those requirements at night local time (when they are more likely to be charging, on free wi-fi, and idle), and so there may be significant diurnal patterns in device availability. Further, because local time of day corresponds directly to longitude, this introduces a strong geographic bias in the source of the data. Eichner *et al.* [92] described this issue and some mitigation strategies, but many open questions remain.

Dataset Shift Finally, we note that the temporal dependence of the distributions $\mathcal{Q}$ and $\mathcal{P}$ may introduce dataset shift in the classic sense (differences between the train and test distributions). Furthermore, other criteria may make the set of clients eligible to train a federated model different from the set of clients where that model will be deployed. For example, training may require devices with more memory than is needed for inference. These issues are explored in more depth in Section 6. Adapting techniques for handling dataset shift to federated learning is another interesting open question.

3.1.1 Strategies for Dealing with Non-IID Data

The original goal of federated learning, training a single global model on the union of client datasets, becomes harder with non-IID data. One natural approach is to modify existing algorithms (e.g., through different hyperparameter choices) or develop new ones in order to more effectively achieve this objective. This approach is considered in Subsection 3.2.2.

For some applications, it may be possible to augment data in order to make the data across clients more similar. One approach is to create a small dataset which can be shared globally. This dataset may originate from a publicly available proxy data source, a separate dataset from the clients' data which is not privacy sensitive, or perhaps a distillation of the raw data following Wang *et al.* [93].

The heterogeneity of client objective functions gives additional importance to the question of how to craft the objective function—it is no-longer clear that treating all examples equally makes sense. Alternatives include limiting the contributions of the data from any one user (which is also important for privacy, see Section 4) and introducing other notions of fairness among the clients; see discussion in Section 6.

But if we have the capability to run training on the local data on each device (which is necessary for federated learning of a global model), is training a single global model even the right goal? There are many cases where having a single model is to be preferred, e.g., in order to provide a model to clients with no data, or to allow manual validation and quality assurance before deployment. Nevertheless, since local training is possible, it becomes feasible for each client to have a customized model. This approach can turn the non-IID problem from a bug to a feature, almost literally—since each client has its own model, the client's identity effectively parameterizes the model, rendering some pathological but degenerate non-IID distributions trivial. For example, if for each i, $\mathcal{P}_i(y)$ has support on only a single label, finding a high-accuracy global model may be very challenging (especially if x is relatively uninformative), but training a high-accuracy local model is trivial (only a constant prediction is needed). Such multi-model approaches are considered in depth in Subsection 3.3. In addition to addressing non-identical client distributions, using a plurality of models can also address violations of independence stemming from changes in client availability. For example, the approach of Eichner *et al.* [92] uses a single training run but averages different iterates in order to provide different models for inference based on the timezone/longitude of clients.

3.2 Optimization Algorithms for Federated Learning

In prototypical federated learning tasks, the goal is to learn a single global model that minimizes the empirical risk function over the entire training dataset, that is, the union of the data across all the clients. The main difference between federated optimization algorithms and standard distributed training methods is the need to address the characteristics of Table 1.1—for optimization, non-IID and unbalanced data, limited communication bandwidth, and unreliable and limited device availability are particularly salient.

FL settings where the total number of devices is huge (e.g., across mobile devices) necessitate algorithms that only require a handful of clients to participate per round (client sampling). Further, each device is likely to participate no more than once in the training of a given model, so stateless algorithms are necessary. This rules out the direct application of a variety of approaches that are quite effective in the datacenter context, for example stateful optimization algorithms like ADMM, and stateful compression strategies that modify updates based on residual compression errors from previous rounds.

Another important practical consideration for federated learning algorithms is composability with other techniques. Optimization algorithms do not run in isolation in a production deployment, but need to be combined with other techniques like cryptographic secure aggregation protocols (Subsection 4.2.1), differential privacy (DP) (Subsection 4.2.2), and model and update compression (Subsection 3.5). As noted in Subsection 1.1.2, many of these techniques can be applied to primitives like "`sum over selected clients`" and "`broadcast to selected clients`", and so expressing optimization algorithms in terms of these primitives provides a valuable separation of concerns, but may also exclude certain techniques such as applying updates asynchronously.

One of the most common approaches to optimization for federated learning is the Federated Averaging algorithm [1], an adaption of local-update or parallel SGD.[2] Here, each client runs some number of SGD steps locally, and then the updated local models are averaged to form the updated global model on the coordinating server. Pseudocode is given in Algorithm 1.

Performing local updates and communicating less frequently with the central server addresses the core challenges of respecting data locality

[2]Federated Averaging applies local SGD to a randomly sampled subset of clients on each round, and proposes a specific update weighting scheme.

constraints and of the limited communication capabilities of mobile device clients. However, this family of algorithms also poses several new algorithmic challenges from an optimization theory point of view. In Subsection 3.2, we discuss recent advances and open challenges in federated optimization algorithms for the cases of IID and non-IID data distribution across the clients respectively. The development of new algorithms that specifically target the characteristics of the federated learning setting remains an important open problem.

Algorithm 1 Federated Averaging (local SGD), when all clients have the same amount of data.

```
Server executes:
  initialize $x_0$
  for each round $t = 1, 2, \ldots,$ T do
    $S_t \leftarrow$ (random set of $M$ clients)
    for each client $i \in S_t$ in parallel do
      $x^i_{t+1} \leftarrow \text{ClientUpdate}(i, x_t)$
    $x_{t+1} \leftarrow \sum_{k=1}^{M} \frac{1}{M} x^i_{t+1}$

ClientUpdate($i, x$):
  for local step $j = 1, \ldots, K$ do
    $x \leftarrow x - \eta \nabla f(x; z)$ for $z \sim \mathcal{P}_i$
  return $x$ to server
```

3.2.1 Optimization Algorithms and Convergence Rates for IID Datasets

While a variety of different assumptions can be made on the per-client functions being optimized, the most basic split is between assuming IID and non-IID data. Formally, having IID data at the clients means that each mini-batch of data used for a client's local update is statistically identical to a uniformly drawn sample (with replacement) from the entire training dataset (the union of all local datasets at the clients). Since the clients independently collect their own training data which vary in both size and distribution, and these data are not shared with other clients or the central node, the IID assumption clearly almost never holds in practice. However, this assumption greatly simplifies theoretical convergence analysis of federated optimization algorithms, as well as establishes a baseline that can be used to understand the impact of non-IID data on optimization rates. Thus, a natural first step is to obtain an understanding of the landscape of optimization algorithms for the IID data case.

Formally, for the IID setting let us standardize the stochastic optimization problem

$$\min_{x \in \mathbb{R}^m} F(x) := \mathop{\mathbb{E}}_{z \sim \mathcal{P}}[f(x; z)].$$

We assume an intermittent communication model as in e.g., Woodworth *et al.* [94, Subsection 4.4], where M stateless clients participate in each of T rounds, and during each round, each client can compute gradients for K samples (e.g., minibatches) $z_1, \ldots, z_K$ sampled IID from $\mathcal{P}$ (possibly using these to take sequential steps). In the IID-data setting clients are interchangeable, and we can without loss of generality assume $M = N$. Table 3.1 summarizes the notation used in this section.

Different assumptions on f will produce different guarantees. We will first discuss the convex setting and later review results for non-convex problems.

Baselines and State-of-the-Art for Convex Problems In this section we review convergence results for H-smooth, convex (but not necessarily strongly convex) functions under the assumption that the variance of the stochastic gradients is bounded by σ^2. More formally, by H-smooth we mean that for all z, $f(\cdot; z)$ is differentiable and has a H-Lipschitz

Table 3.1: Notation for the discussion of FL algorithms including federated averaging

N	Total number of clients
M	Clients per round
T	Total communication rounds
K	Local steps per round

gradient, that is, for all choices of x, y

$$\|\nabla f(x, z) - \nabla f(y, z)\| \leq H\|x - y\|.$$

We also assume that for all x, the stochastic gradient $\nabla_x f(x; z)$ satisfies

$$\mathop{\mathbb{E}}_{z \sim \mathcal{P}} \|\nabla_x f(x; z) - \nabla F(x)\| \leq \sigma^2.$$

When analyzing the convergence rate of an algorithm with output x_T after T iterations, we consider the term

$$\mathbb{E}[F(x_T)] - F(x^*) \tag{3.1}$$

where $x^* = \arg\min_x F(x)$. All convergence rates discussed herein are upper bounds on this term. A summary of convergence results for such functions is given in Table 3.2.

Federated averaging (a.k.a. parallel SGD/local SGD) competes with two natural baselines: First, we may keep x fixed in local updates during each round, and compute a total of KM gradients at the current x, in order to run accelerated minibatch SGD. Let $\bar{x}$ denote the average of T iterations of this algorithm. We then have the upper bound

$$\mathcal{O}\left(\frac{H}{T^2} + \frac{\sigma}{\sqrt{TKM}}\right)$$

for convex objectives [95]–[97]. Note that the first expectation is taken with respect to the randomness of z in the training procedure as well.

A second natural baseline is to ignore all but 1 of the M active clients, which allows (accelerated) sequential SGD to execute for KT steps. Applying the same general bounds cited above, this approach offers an upper bound of

$$\mathcal{O}\left(\frac{H}{(TK)^2} + \frac{\sigma}{\sqrt{TK}}\right).$$

Comparing these two results, we see that minibatch SGD attains the optimal "statistical" term ($\sigma/\sqrt{TKM}$), whilst SGD on a single device (ignoring the updates of the other devices) achieves the optimal "optimization" term ($H/(TK)^2$).

The convergence analysis of local-update SGD methods is an active current area of research [53], [98], [99], [102]–[106]. The first convergence

Table 3.2: Convergence rates for a (non-comprehensive) set of distributed optimization algorithms in the IID-data setting. We assume M devices participate in each iterations, and the loss functions are H-smooth, convex, and we have access to stochastic gradients with variance at most σ^2. All rates are upper bounds on (3.1) after T iterations (potentially with some iterate averaging scheme)

Method	Comments	Convergence
Baselines		
Mini-batch SGD	Batch size KM	$\mathcal{O}\left(\frac{H}{T} + \frac{\sigma}{\sqrt{TKM}}\right)$
SGD	(on one worker, no communication)	$\mathcal{O}\left(\frac{H}{TK} + \frac{\sigma}{\sqrt{TK}}\right)$
Baselines with acceleration[a]		
A-mini-batch SGD [95], [97]	Batch size KM	$\mathcal{O}\left(\frac{H}{T^2} + \frac{\sigma}{\sqrt{TKM}}\right)$
A-SGD [97]	(on one worker, no communication)	$\mathcal{O}\left(\frac{H}{(TK)^2} + \frac{\sigma}{\sqrt{TK}}\right)$
Parallel SGD/Fed-Avg/Local SGD		
Yu *et al.* [98],[b] Stich [99][c]	Gradient norm bounded by G	$\mathcal{O}\left(\frac{HKM}{T}\frac{G^2}{\sigma^2} + \frac{\sigma}{\sqrt{TKM}}\right)$
Wang and Joshi [53],[b] Stich and Karimireddy [100]		$\mathcal{O}\left(\frac{HM}{T} + \frac{\sigma}{\sqrt{TKM}}\right)$
Other algorithms		
SCAFFOLD [101]	Control variates and two stepsizes	$\mathcal{O}\left(\frac{H}{T} + \frac{\sigma}{\sqrt{TKM}}\right)$

[a]There are no accelerated fed-avg/local SGD variants so far.
[b]These papers consider the smooth non-convex setting, we adapt here the results for our setting.
[c]This paper considers the smooth strongly convex setting, we adapt here the results for our setting.

results for local-update SGD methods were derived under the bounded gradient norm assumption in Stich [99] for strongly-convex and in Yu *et al.* [98] for non-convex objective functions. These analyses could attain the desired $\sigma/\sqrt{TKM}$ statistical term with suboptimal optimization term (in Table 3.2 we summarize these results for the middle ground of convex functions).

By removing the bounded gradient assumption, Wang and Joshi [53] and Stich and Karimireddy [100] could further improve the optimization term to HM/T. These result show that if the number of local steps K is smaller than T/M^3 then the (optimal) statistical term is dominating the rate. However, for typical cross-device applications we might have $T = 10^6$ and $M = 100$ (Table 1.2), implying $K = 1$.

Often in the literature the convergence bounds are accompanied by a discussion on how large K may be chosen in order to reach asymptotically the same statistical term as the convergence rate of mini-batch SGD. For strongly convex functions, this bound was improved by Khaled *et al.* [102] and further in Stich and Karimireddy [100].

For non-convex objectives, Yu *et al.* [98] showed that local SGD can achieve asymptotically an error bound $1/\sqrt{TKM}$ if the number of local updates K are smaller than $T^{1/3}/M$. This convergence guarantee was further improved by Wang and Joshi [53] who removed the bounded gradient norm assumption and showed that the number of local updates can be as large as T/M^3. The analysis in [53] can also be applied to other algorithms with local updates, and thus yields the first convergence guarantee for decentralized SGD with local updates (or periodic decentralized SGD) and elastic averaging SGD [107]. Haddadpour *et al.* [108] improves the bounds in Wang and Joshi [53] for functions satisfying the Polyak–Lojasiewicz (PL) condition [109], a generalization of strong convexity. In particular, Haddadpour *et al.* [108] show that for PL functions, T^2/M local updates per round leads to a $\mathcal{O}(1/TKM)$ convergence.

While the above works focus on convergence as a function of the number of iterations performed, practitioners often care about wall-clock convergence speed. Assessing this must take into account the effect of the design parameters on the time spent per iteration based on the relative cost of communication and local computation. Viewed in this light, the focus on seeing how large K can be while maintaining the statistical rate may not be the primary concern in federated learning, where one may assume almost infinite datasets (very large N). The costs (at least in wall-clock time) are small for increasing M, and so it may be more natural to increase M sufficiently to match the optimization term, and then tune K to maximize wall-clock optimization performance. How then to choose K? Performing more local updates at the clients will increase the divergence between the resulting local models at the clients, before they are averaged. As a result, the error convergence in terms of training loss versus the total number of sequential SGD steps TK is slower. However, performing more local updates saves significant communication cost and reduces the time spent per iteration. The optimal number of local updates strikes a balance between these two phenomena and achieves the fastest error versus wallclock time convergence. Wang and Joshi [110] propose an adaptive communication strategy that adapts K according to the training loss at regular intervals during the training.

Another important design parameter in federated learning is the model aggregation method used to update the global model using the updates made by the selected clients. In the original federated learning paper, McMahan *et al.* [1] proposes taking a weighted average of the local models, in proportion to the size of local datasets. For IID data, where each client is assumed to have a infinitely large dataset, this reduces to taking a simple average of the local models. However, it is unclear whether this aggregation method will result in the fastest error convergence.

There are many open questions in federated optimization, even with IID data. Woodworth *et al.* [94] highlights several gaps between upper and lower bounds for optimization relevant to the federated learning setting, particularly for "intermittent communication graphs", which captures local SGD approaches, but convergence rates for such approaches are not known to match the corresponding lower bounds. In Table 3.2 we highlight convergence results for the convex setting. Whilst most schemes are able to reach the asymptotically dominant statistical term, none are able to match the convergence rate of accelerated mini-batch SGD. It is an open problem if federated averaging algorithms can close this gap.

Local-update SGD methods where all M clients perform the same number of local updates may suffer from a common scalability issue—they can be bottlenecked if any one client unpredictably slows down or fails. Several approaches for dealing with this are possible, but it is far from clear which are optimal, especially when the potential for bias is considered (see Section 6). Bonawitz *et al.* [32] propose over-provisioning clients (e.g., request updates from $1.3M$ clients), and then accepting the first M updates received and rejecting updates from stragglers. A slightly more sophisticated solution is to fix a time window and allow clients to perform as many local updates K_i as possible within this time, after which their models are averaged by a central server. Wang *et al.* [111] analyzed the computational heteogeneity introduced by this approach in theory. An alternative method to overcome the problem of straggling clients is to fix the number of local updates at τ, but allow clients to update the global model in an asynchronous or lock-free fashion. Although some previous works [50], [107], [112] have proposed similar methods, the error convergence analysis is an open and challenging problem. A larger challenge in the FL setting, however, is that as discussed at the beginning of Subsection 3.2, asynchronous approaches may be difficult to combine with complimentary techniques like differential privacy or secure aggregation.

Besides the number of local updates, the choice of the size of the set of clients selected per training round presents a similar trade-off as the number of local updates. Updating and averaging a larger number of client models per training round yields better convergence, but it makes the training vulnerable to slowdown due to unpredictable tail delays in computation/communication at/with the clients.

The analysis of local SGD/Federated Averaging in the non-IID setting is even more challenging; results and open questions related to this are considered in the next section, along with specialized algorithms which directly address the non-IID problem.

3.2.2 Optimization Algorithms and Convergence Rates for Non-IID Datasets

In contrast to well-shuffled mini-batches consisting of independent and identically distributed (IID) examples in centralized learning, federated learning uses local data from end user devices, leading to many varieties of non-IID data (Subsection 3.1).

In this setting, each of N clients has a local data distribution $\mathcal{P}_i$ and a local objective function

$$f_i(x) = \mathop{\mathbb{E}}_{z \sim \mathcal{P}_i} [f(x; z)]$$

where we recall that $f(x;z)$ is the loss of a model x at an example z. We typically wish to minimize

$$F(x) = \frac{1}{N}\sum_{i=1}^{N} f_i(x). \tag{3.2}$$

Note that we recover the IID setting when each $\mathcal{P}_i$ is identical. We will let F^* denote the minimum value of F, obtained the point x^*. Analogously, we will let f_i^* denote the minimum value of f_i.

As in the IID setting, we assume an intermittent communication model (e.g., Woodworth *et al.* [94, Subsection 4.4]), where M stateless clients participate in each of T rounds, and during each round, each client can compute gradients for K samples (e.g., minibatches). The difference here is that the samples $z_{i,1},\ldots,z_{i,K}$ sampled at client i are drawn from the client's local distribution $\mathcal{P}_i$. Unlike the IID setting, we cannot necessarily assume $M = N$, as the client distributions are not all equal. In the following, if an algorithm relies on $M = N$, we will omit M and simply write N. We note that while such an assumption may be compatible with the cross-silo federated setting in Table 1.1, it is generally infeasible in the cross-device setting.

While [53], [98]–[100] mainly focused on the IID case, the analysis technique can be extended to the non-IID case by adding an assumption on data dissimilarities, for example by constraining the difference between client gradients and the global gradient [36], [49], [111], [113], [114] or the difference between client and global optimum values [115], [116]. Under this assumption, Yu *et al.* [117] showed that the error bound of local SGD in the non-IID case becomes worse. In order to achieve the rate of $1/\sqrt{TKN}$ (under non-convex objectives), the number of local updates K should be smaller than $T^{1/3}/N$, instead of T/N^3 as in the IID case [53]. Li *et al.* [113] proposed to add a proximal term in each local objective function so as to make the algorithm be more robust to the heterogeneity across local objectives. The proposed FedProx algorithm empirically improves the performance of federated averaging. Khaled *et al.* [115] assumes all clients participate, and uses batch gradient descent on clients, which can potentially converge faster than stochastic gradients on clients.

Recently, a number of works have made progress in relaxing the assumptions necessary for analysis so as to better apply to practical uses of Federated Averaging. For example, Li *et al.* [116] studied the convergence of Federated Averaging in a more realistic setting where only a subset of clients are involved in each round. In order to guarantee the convergence, they assumed that the clients are selected either uniformly at random or with probabilities that are in proportion to the sizes of local datasets. Nonetheless, in practice the server may not be able to sample clients in these idealized ways—in particular, in cross-device settings only devices that meet strict eligibility requirements (e.g., charging, idle, free WiFi) will be selected to participate in the computation. At different times within a day, the clients characteristics can vary significantly. Eichner *et al.* [92] formulated this problem and studied the convergence of semi-cyclic SGD, where multiple blocks of clients with different characteristics are sampled from following a regular cyclic pattern (e.g., diurnal). Clients can perform different local steps because of heterogeneity in their computing capacities. Wang *et al.* [111] proves that FedAvg and many other federated learning algorithms will converge to the stationary points of a mismatched objective function in the presence of heterogeneous local steps. They refer to this problem as *objective inconsistency* and propose a simple technique to eliminate the inconsistency problem from federated learning algorithms.

We summarize recent theoretical results in Table 3.3. All the methods in Table 3.3 assume smoothness or Lipschitz gradients for the local functions on clients. The error bound is measured by optimal objective (3.1) for convex functions and norm of gradient for nonconvex functions. For each method, we present the key non-IID assumption, assumptions on each client function $f_i(x)$, and other auxiliary assumptions. We also briefly describe each method as a variant of the federated averaging algorithm, and show the simplified convergence rate eliminating constants. Assuming the client functions are strongly convex could help the convergence rate [101], [116]. Bounded gradient variance, which is a widely used assumption to analyze stochastic gradient methods, is often used when clients use stochastic local updates [36], [49], [101], [114], [116]. Li *et al.* [116] directly analyzes the Federated Averaging algorithm, which applies K steps of local updates on randomly sampled

Table 3.3: Convergence rates for a (non-comprehensive) set of federated optimization methods in non-IID settings. We summarize the key assumptions for non-IID data, local functions on each client, and other assumptions. We also present the variant of the algorithm comparing to Federated Averaging and the convergence rates that eliminate constant

Non-IID Assumptions

Symbol	Full Name	Explanation
BCGV	Bounded inter-client gradient variance	$\mathbb{E}_i \lVert \nabla f_i(x) - \nabla F(x) \rVert^2 \leq \eta^2$
BOBD	Bounded optimal objective difference	$F^* - \mathbb{E}_i[f_i^*] \leq \eta^2$
BOGV	Bounded optimal gradient variance	$\mathbb{E}_i \lVert \nabla f_i(x^*) \rVert^2 \leq \eta^2$
BGV	Bounded gradient dissimilarity	$\mathbb{E}_i \lVert \nabla f_i(x) \rVert^2 / \lVert \nabla F(x) \rVert^2 \leq \eta^2$

Other Assumptions and Variants

Symbol	Explanation
CVX	Each client function $f_i(x)$ is convex.
SCVX	Each client function $f_i(x)$ is μ-strongly convex.
BNCVX	Each client function has bounded nonconvexity with $\nabla^2 f_i(x) \succeq -\mu I$.
BLGV	The variance of stochastic gradients on local clients is bounded.
BLGN	The norm of any local gradient is bounded.
LBG	Clients use the full batch of local samples to compute updates.
Dec	Decentralized setting, assumes the connectivity of network is good.
AC	All clients participate in each round.
1step	One local update is performed on clients in each round.
Prox	Use proximal gradient steps on clients.
VR	Variance reduction which needs to track the state.

Convergence Rates

Method	Non-IID	Other Assumptions	Variant	Rate
Lian *et al.* [36]	BCGV	BLGV	Dec; AC; 1step	$O(1/T) + O(1/\sqrt{NT})$
PD-SGD [114]	BCGV	BLGV	Dec; AC	$O(N/T) + O(1/\sqrt{NT})$
MATCHA [49]	BCGV	BLGV	Dec	$O(1/\sqrt{TKM}) + O(M/KT)$
Khaled *et al.* [115]	BOGV	CVX	AC; LBG	$O(N/T) + O(1/\sqrt{NT})$
Li *et al.* [116]	BOBD	SCVX; BLGV; BLGN	–	$O(K/T)$
FedProx [113]	BGV	BNCVX	Prox	$O(1/\sqrt{T})$
SCAFFOLD [101]	–	SCVX; BLGV	VR	$O(1/TKM) + O(e^{-T})$

M clients in each round, and presents a rate that suggests local updates ($K > 1$) could slow down the convergence. Clarifying the regimes where $K > 1$ may hurt or help convergence is an important open problem.

Connections to Decentralized Optimization The objective function of federated optimization has been studied for many years in the decentralized optimization community. As first shown in Wang and Joshi [53], the convergence analysis of decentralized SGD can be applied to or combined with local SGD with a proper setting of the network topology matrix (mixing matrix). In order to reduce the communication overhead, Wang and Joshi [53] proposed periodic decentralized SGD (PD-SGD) which allows decentralized SGD to have multiple local updates as Federated Averaging. This algorithm is extended by Li *et al.* [114] to the non-IID case. MATCHA [49] further improves the performance of PD-SGD by randomly sampling clients for computation and communication, and provides a convergence analysis showing that local updates can accelerate convergence.

Acceleration, Variance Reduction and Adaptivity Momentum, variance-reduction, and adaptive learning rates are all promising techniques to improve convergence and generalization of first-order methods. However, there is no single manner in which to incorporate these techniques into FedAvg. SCAFFOLD [101] models the difference in client updates using control variates to perform variance reduction. Notably, this allows convergence results not relying on bounding the amount of heterogeneity among clients. As for momentum, Yu *et al.* [117] propose allowing each client to maintain a local momentum buffer and average the local buffers and the local model parameters at each communication round. Although this method empirically improves the final accuracy of local SGD, this doubles the per-round communication cost. A similar scheme is used by Xie *et al.* [118] to design a variant of local SGD in which clients locally perform Adagrad [119], [120]. Reddi *et al.* [121] instead proposes using adaptive learning rates at the server-level, developing federated versions of adaptive optimization methods with the same communication cost as FedAvg. This framework generalizes the server momentum framework proposed by Hsu *et al.* [122], Wang *et al.* [123], which allows momentum without increasing communication costs. While both [117], [123] showed that the momentum variants of local SGD can converge to stationary points of non-convex objective functions at the same rate as synchronous mini-batch SGD, it is challenging to prove momentum accelerates the convergence rate in the federated learning setting. Recently, Karimireddy *et al.* [124] proposed a general approach for adapting centralized optimization algorithms to the heterogeneous federated setting (MIME framework and algorithms).

3.3 Multi-Task Learning, Personalization, and Meta-Learning

In this section we consider a variety of "multi-model" approaches—techniques that result in effectively using different models for different clients at inference time. These techniques are particularly relevant when faced with non-IID data (Subsection 3.1), since they may outperform even the best possible shared global model. We note that personalization has also been studied in the fully decentralized setting [35], [39], [54], [125], where training individual models is particularly natural.

3.3.1 Personalization via Featurization

The remainder of this section specifically considers techniques that result in different users running inference with different model parameters (weights). However, in some applications similar benefits can be achieved by simply adding user and context features to the model. For example, consider a language model for next-word-prediction in a mobile keyboard as in Hard *et al.* [11]. Different clients are likely to use language differently, and in fact on-device personalization of model parameters has yielded significant improvements for this problem [126]. However, a complimentary approach may be to train a federated model that takes as input not only the words the user has typed so far, but a variety of other user and context features—What words does this user frequently use? What app are they currently using? If they are chatting, what messages have they sent to this person before? Suitably featurized, such inputs can allow a shared global model to produce highly personalized predictions. However, largely because few public datasets contain such auxiliary features, developing model architectures that can effectively incorporate context information for different tasks remains an important open problem with the potential to greatly increase the utility of FL-trained models.

3.3.2 Multi-Task Learning

If one considers each client's local problem (the learning problem on the local dataset) as a separate task (rather than as a shard of a single partitioned dataset), then techniques from multi-task learning [127] immediately become relevant. Notably, Smith *et al.* [128] introduced the MOCHA algorithm for multi-task federated learning, directly tackling challenges of communication efficiency, stragglers, and fault tolerance. In multi-task learning, the result of the training process is one model per task. Thus, most multi-task learning algorithms assume all clients (tasks) participate in each training round, and also require stateful clients since each client is training an individual model. This makes such techniques relevant for cross-silo FL applications, but harder to apply in cross-device scenarios.

Another approach is to reconsider the relationship between clients (local datasets) and learning tasks (models to be trained), observing that there are points on a spectrum between a single global model and different models for every client. For example, it may be possible to apply techniques from multi-task learning (as well as other approaches like personalization, discussed next), where we take the "task" to be a subset of the clients, perhaps chosen explicitly (e.g., based on geographic region, or characteristics of the device or user), or perhaps based on clustering [129] or the connected components of a learned graph over the clients [54]. The development of such algorithms is an important open problem. See Subsection 4.4.4 for a discussion of how sparse federated learning problems, such as those arising naturally in this type of multi-task problem, might be approached without revealing to which client subset (task) each client belongs.

3.3.3 Local Fine Tuning and Meta-Learning

By local fine tuning, we refer to techniques which begin with the federated training of a single model, and then deploy that model to all clients, where it is personalized by additional training on the local dataset before use in inference. This approach integrates naturally into the typical lifecycle of a model in federated learning (Subsection 1.1.1). Training of the global model can still proceed using only small samples of clients on each round (e.g., 100s); the broadcast of the global model to all clients (e.g., many millions) only happens once, when the model is deployed. The only difference is that before the model is used to make live predictions on the client, a final training process occurs, personalizing the model to the local dataset.

Given a global model that performs reasonably well, what is the best way to personalize it? In non-federated learning, researchers often use fine-tuning, transfer learning, domain adaptation [130]–[134], or interpolation with a personal local model. Of course, the precise technique used for such interpolations is key and it is important to determine its corresponding learning guarantees in the context of federated learning. Further, these techniques often assume only a pair of domains (source and target), and so some of the richer structure of federated learning may be lost.

One approach for studying personalization and non-IID data is via a connection to *meta-learning*, which has emerged as a popular setting for model adaptation. In the standard learning-to-learn (LTL) setup [135], one has a meta-distribution over tasks, samples from which are used to learn a learning algorithm, for example by finding a good restriction of the hypothesis space. This is in fact a good match for the statistical setting discussed in Subsection 3.1, where we sample a client (task) $i \sim \mathcal{Q}$, and then sample data for that client (task) from $\mathcal{P}_i$.

Recently, a class of algorithms referred to as *model-agnostic meta-learning* (MAML) have been developed that meta-learn a global model, which can be used as a starting point for learning a good model adapted to a given task, using only a few local gradient steps [136]. Most notably, the training phase of the popular Reptile algorithm [137] is closely related to Federated Averaging [1]—Reptile allows for a server learning rate and assumes all clients have the same amount of data, but is otherwise the same. Khodak *et al.* [138] and Jiang *et al.* [139] explore the connection between FL and MAML, and show how the MAML setting is a relevant framework to model the personalization objectives for FL. Chai Sim *et al.* [140] applied local fine tuning to personalize speech recognition models in federated learning. Fallah *et al.* [141] developed a new algorithm called Personalized FedAvg by connecting

MAML instead of Reptile to federated learning. Additional connections with differential privacy were studied in [142].

The general direction of combining ideas from FL and MAML is relatively new, with many open questions:

- The evaluation of MAML algorithms for supervised tasks is largely focused on synthetic image classification problems [143], [144] in which infinite artificial tasks can be constructed by subsampling from classes of images. FL problems, modeled by existing datasets used for simulated FL experiments (Appendix A.1), can serve as realistic benchmark problems for MAML algorithms.

- In addition to an empirical study, or optimization results, it would be useful to analyze the theoretical guarantees of MAML-type techniques and study under what assumptions they can be successful, as this will further elucidate the set of FL domains to which they may apply.

- The observed gap between the global and personalized accuracy [139] creates a good argument that personalization should be of central importance to FL. However, none of the existing works clearly formulates what would be comprehensive metrics for measuring personalized performance; for instance, is a small improvement for every client preferable to a larger improvement for a subset of clients? See Subsection 6 for a related discussion.

- Jiang *et al.* [139] highlighted the fact that models of the same structure and performance, but trained differently, can have very different capacity to personalize. In particular, it appears that training models with the goal of maximizing global performance might actually hurt the model's capacity for subsequent personalization. Understanding the underlying reasons for this is a question relevant for both FL and the broader ML community.

- Several challenging FL topics including personalization and privacy have begun to be studied in this multi-task/LTL framework [138], [139], [142]. Is it possible for other issues such as concept drift to also be analyzed in this way, for example as a problem in lifelong learning [145]?

- Can non-parameter transfer LTL algorithms, such as ProtoNets [146], be of use for FL?

3.3.4 When is a Global FL-Trained Model Better?

What can federated learning do for you that local training on one device cannot? When local datasets are small and the data is IID, FL clearly has an edge, and indeed, real-world applications of federated learning [10], [11], [14] benefit from training a single model across devices. On the other hand, given pathologically non-IID distributions (e.g., $\mathcal{P}_i(y \mid x)$ directly disagree across clients), local models will do much better. Thus, a natural theoretical question is to determine under what conditions the shared global model is better than independent per-device models. Suppose we train a model h_k for each client k, using the sample of size m_k available from that client. Can we guarantee that the model h_{FL} learned via federated learning is at least as accurate as h_k when used for client k? Can we quantify how much improvement can be expected via federated leaning? And can we develop personalization strategies with theoretical guarantees that at least match the performance of both natural baselines (h_k and h_{FL})?

Several of these problems relate to previous work on multiple-source adaptation and agnostic federated learning [133], [147]–[149]. The hardness of these questions depends on how the data is distributed among parties. For example, if data is vertically partitioned, each party maintaining private records of different feature sets about common entities, these problems may require addressing record linkage [150] within the federated learning task. Independently of the eventual technical levy of carrying out record linkage privately [151], the task itself happens to be substantially noise prone in the real world [152] and only sparse results have addressed its impact on training models [65]. Techniques for robustness and privacy can make local models relatively stronger, particularly for non-typical clients [153]. Loss factorization tricks can be used in supervised learning to alleviate up to the vertical partition assumption itself, but the practical benefits depend on the distribution of data and the number of parties [154].

3.4 Adapting ML Workflows for Federated Learning

Many challenges arise when adapting standard machine learning workflows and pipelines (including data augmentation, feature engineering, neural architecture design, model selection, hyperparameter optimization, and debugging) to decentralized datasets and resource-constrained mobile devices. We discuss several of these challenges below.

3.4.1 Hyperparameter Tuning

Running many rounds of training with different hyperparameters on resource-constrained mobile devices may be restrictive. For small device populations, this might result in the over-use of limited communication and compute resources. However, recent deep neural networks crucially depend on a wide range of hyperparameter choices regarding the neural network's architecture, regularization, and optimization. Evaluations can be expensive for large models and large-scale on-device datasets. Hyperparameter optimization (HPO) has a long history under the framework of AutoML [155]–[157], but it mainly concerns how to improve the model accuracy [158]–[161] rather than communication and computing efficacy for mobile devices. Therefore, we expect that further research should consider developing solutions for efficient hyperparameter optimization in the context of federated learning.

In addition to general-purpose approaches to the hyperparameter optimization problem, in the training space specifically the development of easy-to-tune optimization algorithms is a major open area. Centralized training already requires tuning parameters like learning rate, momentum, batch size, and regularization. Federated learning adds potentially more hyperparameters—separate tuning of the aggregation/global model update rule and local client optimizer, number of clients selected per round, number of local steps per round, configuration of update compression algorithms, and more. Such hyperparameters can be crucial to obtaining a good trade-off between accuracy and convergence, and may actually impact the quality of the learned model [162]. In addition to a higher-dimensional search space, federated learning often also requires longer wall-clock training times and limited compute resources. These challenges could be addressed by optimization algorithms that are robust to hyperparameter settings (the same hyperparameter values work for many different real world datasets and architectures), as well as adaptive or self-tuning algorithms [163], [164].

3.4.2 Neural Architecture Design

Neural architecture search (NAS) in the federated learning setting is motivated by the drawbacks of the current practice of applying predefined deep learning models: the predefined architecture of a deep learning model may not be the optimal design choice when the data generated by users are invisible to model developers. For example, the neural architecture may have some redundant component for a specific dataset, which may lead to unnecessary computing on devices; there may be a better architectural design for the non-IID data distribution. The approaches to personalization discussed in Subsection 3.3 still share the same model architecture among all clients. The recent progress in NAS [165]–[173] provides a potential way to address these drawbacks. There are three major methods for NAS, which utilize evolutionary algorithms, reinforcement learning, or gradient descent to search for optimal architectures for a specific task on a specific dataset. Among these, the gradient-based method leverages efficient gradient back-propagation with weight sharing, reducing the architecture search process from over 3000 GPU days to only 1 GPU day. Another interesting paper recently published, involving Weight Agnostic Neural Networks [174], claims that neural network architectures alone, without learning any weight parameters, may encode solutions for a given task. If this technique further develops and reaches widespread use, it may be applied to the federated learning without collaborative training among devices. Although these methods have not been developed for distributed settings such as federated learning, they are all feasible to be transferred to the federated setting. Neural Architecture Search (NAS) for a global or personalized model in the federated learning setting is promising, and early exploration has been made in [175].

3.4.3 Debugging and Interpretability for FL

While substantial progress has been made on the federated training of models, this is only part of a complete ML workflow. Experienced modelers often directly inspect subsets of the data for tasks including basic sanity checking, debugging misclassifications, discovering outliers, manually labeling examples, or detecting bias in the training set. Developing privacy-preserving techniques to answer such questions on decentralized data is a major open problem. Recently, Augenstein *et al.* [176] proposed the use of differentially private generative models (including GANs), trained with federated learning, to answer some questions of this type. However, many open questions remain (see discussion in [176]), in particular the development of algorithms that improve the fidelity of FL DP generative models.

3.5 Communication and Compression

It is now well-understood that communication can be a primary bottleneck for federated learning since wireless links and other end-user internet connections typically operate at lower rates than intra- or inter-datacenter links and can be potentially expensive and unreliable. This has led to significant recent interest in reducing the communication bandwidth of federated learning. Methods combining Federated Averaging with sparsification and/or quantization of model updates to a small number of bits have demonstrated significant reductions in communication cost with minimal impact on training accuracy [177]. However, it remains unclear if communication cost can be further reduced, and whether any of these methods or their combinations can come close to providing optimal trade-offs between communication and accuracy in federated learning. Characterizing such fundamental trade-offs between accuracy and communication has been of recent interest in theoretical statistics [56], [178]–[183]. These works characterize the optimal minimax rates for distributed statistical estimation and learning under communication constraints. However, it is difficult to deduce concrete insights from these theoretical works for communication bandwidth reduction in practice as they typically ignore the impact of the optimization algorithm. It remains an open direction to leverage such statistical approaches to inform practical training methods.

Compression Objectives Motivated by the limited resources of current devices in terms of compute, memory and communication, there are several different compression objectives of practical value.

(a) *Gradient compression*[3]—reduce the size of the object communicated from clients to server, which is used to update the global model.

(b) *Model broadcast compression*—reduce the size of the model broadcast from server to clients, from which the clients start local training.

(c) *Local computation reduction*—any modification to the overall training algorithm such that the local training procedure is computationally more efficient.

These objectives are in most cases complementary. Among them, (a) has the potential for the most significant practical impact in terms of total runtime. This is both because clients' connections generally have slower upload than download bandwidth[4]—and thus there is more to be gained, compared to (b)—and because the effects of averaging across many clients can enable more aggressive lossy compression schemes. Usually, (c) could be realized jointly with (a) and (b) by specific methods.

Much of the existing literature applies to the objective (a) [177], [184]–[188]. The impact of (b) on convergence in general has not been studied until very recently; an analysis is presented in [189]. Very few methods intend to address all of (a), (b), and (c) jointly. Caldas *et al.* [190] proposed a practical method by constraining the desired

[3]In this section, we use "gradient compression" to include compression applied to any model update, such as the updates produced by Federated Averaging when clients take multiple gradient steps.

[4]See for instance https://www.speedtest.net/reports/.

model update such that only particular submatrices of model variables are necessary to be available on clients; Hamer *et al.* [191] proposed a communication-efficient federated algorithm for learning mixture weights on an ensemble of pre-trained models, based on communicating only a subset of the models to any one device; He *et al.* [192] utilizes bidirectional and alternative knowledge distillation method to transfer knowledge from many compact DNNs to a dense server DNN, which can reduce the local computational burden at the edge devices.

In cross-device FL, algorithms generally cannot assume any state is preserved on the clients (Table 1.1). However, this constraint would typically not be present in the cross-silo FL setting, where the same clients participate repeatedly. Consequently, a wider set of ideas related to error-correction such as [56], [100], [193]–[196] are relevant in this setting, many of which could address both (a) and (b).

An additional objective is to modify the training procedure such that the *final* model is more compact, or efficient for inference. This topic has received a lot of attention in the broader ML community [197]–[202], but these methods either do not have a straightforward mapping to federated learning, or make the training process more complex which makes it difficult to adopt. Research that simultaneously yields a compact final model, while also addressing the three objectives above, has significant potential for practical impact.

For gradient compression, some existing works [188] are developed in the minimax sense to characterize the worst case scenario. However usually in information theory, the compression guarantees are instance specific and depend on the *entropy* of the underlying distribution [203]. In other words, if the data is easily compressible, they are provably compressed heavily. It would be interesting to see if similar instance specific results can be obtained for gradient compression. Similarly, recent works show that learning a compression scheme in a data-dependent fashion can lead to significantly better compression ratio for the case of data compression [204] as well as gradient compression. It is therefore worthwhile to evaluate these data-dependent compression schemes in the federated settings [205].

Compatibility with Differential Privacy and Secure Aggregation Many algorithms used in federated learning such as Secure Aggregation [206] and mechanisms of adding noise to achieve differential privacy [62], [207] are not designed to work with compressed or quantized communications. For example, straightforward application of the Secure Aggregation protocol of Bonawitz *et al.* [60], Bell *et al.* [208] requires an additional $O(\log M)$ bits of communication for each scalar, where M is the number of clients being summed over, and this may render ineffective the aggressive quantization of updates when M is large (though see [163] for a more efficient approach). Existing noise addition mechanisms assume adding real-valued Gaussian or Laplacian noise on each client, and this is not compatible with standard quantization methods used to reduce communication. We note that several recent works allow biased estimators and would work nicely with Laplacian noise [100], however those would not give differential privacy, as they break independence between rounds. There is some work on adding discrete noise [209], but there is no notion whether such methods are optimal. Joint design of compression methods that are compatible with Secure Aggregation, or for which differential privacy guarantees can be obtained, is thus a valuable open problem.

Wireless-FL co-Design The existing literature in federated learning usually neglects the impact of wireless channel dynamics during model training, which potentially undermines both training latency and thus reliability of the entire production system. In particular, wireless interference, noisy channels and channel fluctuations can significantly hinder the information exchange between the server and clients (or directly between individual clients, as in the fully decentralized case, see Subsection 2.1). This represents a major challenge for mission-critical applications, rooted in latency reduction and reliability enhancements. Potential solutions to address this challenge include federated distillation (FD), in which workers exchange their model output parameters (logits) as opposed to the model parameters (gradients and/weights), and optimizing workers' scheduling policy with appropriate communication and computing resources [210]–[212]. Another solution is to leverage the unique characteristics of wireless channels (e.g., broadcast and superposition) as natural data aggregators, in which the simultaneously transmitted analog-waves by different workers are superposed at the server and weighed by the wireless channel coefficients [213]. This yields faster model aggregation at the server, and faster training by a factor up to the number of workers. This is in sharp contrast with the traditional orthogonal frequency division multiplexing (OFDM) paradigm, whereby workers upload their models over orthogonal frequencies whose performance degrades with increasing number of workers [214].

3.6 Application to More Types of Machine Learning Problems and Models

To date, federated learning has primarily considered supervised learning tasks where labels are naturally available on each client. Extending FL to other ML paradigms, including reinforcement learning, semi-supervised and unsupervised learning, active learning, and online learning [45], [215] all present interesting and open challenges.

Another important class of models, highly relevant to FL, are those that can characterize the uncertainty in their predictions. Most modern deep learning models cannot represent their uncertainty nor allow for a probability interpretation of parametric learning. This has motivated recent developments of tools and techniques combining Bayesian models with deep learning. From a probability theory perspective, it is unjustifiable to use single point-estimates for classification. Bayesian neural networks [216] have been proposed and shown to be far more robust to over-fitting, and can easily learn from small datasets. The Bayesian approach further offers uncertainty estimates via its parameters in form of probability distributions, thus preventing over-fitting. Moreover, appealing to probabilistic reasoning, one can predict how the uncertainty can decrease, allowing the decisions made by the network to become more deterministic as the data size grows.

Since Bayesian methods gave us tools to reason about deep models' confidence and also achieve state-of-the-art performance on many tasks, one expects Bayesian methods to provide a conceptual improvement to the classical federated learning. In fact, preliminary work from Lalitha *et al.* [217] shows that incorporating Bayesian methods allows for model aggregation across non-IID data and heterogeneous platforms. However, many questions regarding scalability and computational feasibility have to be addressed.

3.7 Executive Summary

Efficient and effective federated learning algorithms face different challenges compared to centralized training in a datacenter.

- Non-IID data due to non-identical client distributions, violation of independence, and dataset drift (Subsection 3.1) pose a key challenge. Though various methods have been surveyed and discussed in this section, defining and dealing with non-IID data remains an open problem and one of the most active research topics in federated learning.
- Optimization algorithms for federated learning are analyzed in Subsection 3.2 under different settings, e.g., convex and nonconvex functions, IID and non-IID data. Theoretical analysis has proven difficult for the parallel local updates commonly used in federated optimization, and often strict assumptions have to be made to constrain the client heterogeneity. Currently, known convergence rates do not fully explain the empirically-observed effectiveness of the Federated Averaging algorithm over methods such as mini-batch SGD [106].
- Client-side personalization and “multi-model” approaches (Subsection 3.3) can address data heterogeneity and give hope of surpassing the performance of the best fixed global model. Simple personalization methods like fine-tuning can be effective, and offer intrinsic privacy advantages. However, many theoretical and empirical questions remain open: when is a global model better? How many models are necessary? Which federated optimization algorithms combine best with local fine-tuning?
- Adapting centralized training workflows such as hyper-parameter tuning, neural architecture design, debugging, and interpretability

tasks to the federated learning setting (Subsection 3.4) present roadblocks to the widespread adoption of FL in practical settings, and hence constitute important open problems.

- While there has been significant work on communication efficiency and compression for FL (Subsection 3.5), it remains an important and active area. In particular, fully automating the process of enabling compression without impacting convergence for a wide class of models is an important practical goal. Relatively new directions on the theoretical study of communication, compatibility with privacy methods, and co-design with wireless infrastructure are discussed.

- There are many open questions in extending federated learning from supervised tasks to other machine learning paradigms including reinforcement learning, semi-supervised and unsupervised learning, active learning, and online learning (Subsection 3.6).

4

Preserving the Privacy of User Data

Machine learning workflows involve many actors functioning in disparate capacities. For example, users may generate training data through interactions with their devices, a machine learning training procedure extracts cross-population patterns from this data (e.g., in the form of trained model parameters), the machine learning engineer or analyst may assess the quality of this trained model, and eventually the model may be deployed to end users in order to support specific user experiences (see Figure 4.1 next page).

In an ideal world, each actor in the system would learn nothing more than the information needed to play their role. For example, if an analyst only needs to determine whether a particular quality metric exceeds a desired threshold in order to authorize deploying the model to end users, then in an idealized world, that is the only bit of information that would be available to the analyst; such an analyst would need access to neither the training data nor the model parameters, for instance. Similarly, end users enjoying the user experiences powered by the trained model might only require predictions from the model and nothing else.

Furthermore, in an ideal world every participant in the system would be able to reason easily and accurately about what personal information

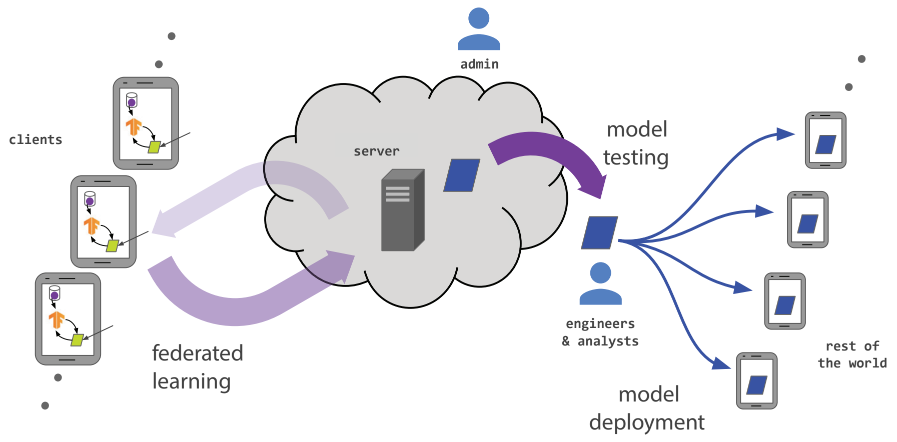


Figure 4.1: The lifecycle of an FL-trained model and the various actors in a federated learning system. (repeated from page 10).

about themselves and others might be revealed by their participation in the system, and participants would be able to use this understanding to make informed choices about how and whether to participate at all.

Producing a system with all of the above ideal privacy properties would be a daunting feat on its own, and even more so while also guaranteeing other desirable properties such as ease of use for all participants, the quality and fairness of the end user experiences (and the models that power them), the judicious use of communication and computation resources, resilience against attacks and failures, and so on.

Rather than allowing perfect to be the enemy of good, we advocate a strategy wherein the overall system is composed of modular units which can be studied and improved relatively independently, while also reminding ourselves that we must, in the end, measure the privacy properties of the complete system against our ideal privacy goals set out above. The open questions raised throughout this section will highlight areas wherein we do not yet understand how to simultaneously achieve all of our goals, either for an individual module or for the system as a whole.

Federated learning provides an attractive structure for decomposing the overall machine learning workflow into the approachable modular units we desire. One of the primary attractions of the federated learning model is that it can provide a level of privacy to participating users through data minimization: the raw user data never leaves the device, and only updates to models (e.g., gradient updates) are sent to the central server. These model updates are more focused on the learning task at hand than is the raw data (i.e., they contain strictly no additional information about the user, and typically significantly less, compared to the raw data), and the individual updates only need to be held ephemerally by the server.

While these features can offer significant practical privacy improvements over centralizing all the training data, there is still no formal guarantee of privacy in this baseline federated learning model. For instance, it is possible to construct scenarios in which information about the raw data is leaked from a client to the server, such as a scenario where knowing the previous model and the gradient update from a user would allow one to infer a training example held by that user. Therefore, this section surveys existing results and outlines open challenges towards designing federated learning systems that can offer rigorous privacy guarantees. We focus on questions specific to the federated learning and analytics setting and leave aside questions that also arise in more general machine learning settings as surveyed in [218].

Beyond attacks targeting user privacy, there are also other classes of attacks on federated learning; for example, an adversary might attempt to prevent a model from being learned at all, or they might attempt to bias the model to produce inferences that are preferable to the adversary. We defer consideration of these types of attacks to Subsection 5.

The remainder of this section is organized as follows. Subsection 4.1 discusses various threat models against which we wish to give protections. Subsection 4.2 lays out a set of core tools and technologies that can be used towards providing rigorous protections against the threat models discussed in Subsection 4.1. Subsection 4.3 assumes the existence of a trusted server and discusses the open problems and challenges in providing protections against adversarial clients and/or analysts. Subsection 4.4 discusses the open problems and challenges in the absence of a fully trusted server. Finally, Subsection 4.5 discusses open questions around user perception.

4.1 Actors, Threat Models, and Privacy in Depth

A formal treatment of privacy risks in FL calls for a holistic and interdisciplinary approach. While some of the risks can be mapped to technical privacy definitions and mitigated with existing technologies, others are more complex and require cross-disciplinary efforts.

Privacy is not a binary quantity, or even a scalar one. This first step towards such formal treatment is a careful characterization of the different actors (see Figure 1.1 from Section 1, repeated on page 61 for convenience) and their roles to ultimately define relevant threat models (see Table 4.1). Thus, for instance, it is desirable to distinguish the view of the server administrator from the view of the analysts that consume the learned models, as it is conceivable that a system that is designed to offer strong privacy guarantees against a malicious analyst may not provide any guarantees with respect to a malicious server. These actors map well onto the threat models discussed elsewhere in the literature; for example, in Bittau *et al.* [219, Subsection 3.1], where the "encoder" corresponds to the client, the "shuffler" generally corresponds to the server, the "analyzer" may correspond to the server or post-processing done by the analyst.

As an example, a particular system might offer a differential privacy[1] guarantee with a particular parameter ε to the view of the server administrator, while the results observed by analysts might have a higher protection $\varepsilon' < \varepsilon$.

Furthermore, it is possible that this guarantee holds only against adversaries with particular limits on their capabilities, e.g., an adversary that can observe everything that happens on the server (but cannot influence the server's behavior) while simultaneously controlling up to a fraction γ of the clients (observing everything they see and influencing their behavior in arbitrary ways); the adversary might also be assumed to be unable to break cryptographic mechanisms instantiated at a particular security level σ. Against an adversary whose strength *exceeds* these limits, the view of the server administrator might still have some differential privacy, but at weaker level $\varepsilon_0 > \varepsilon$.

[1] Differential privacy will be formally introduced in Subsection 4.2.2. For now, it suffices to know that lower ε corresponds with higher privacy.

Table 4.1: Various threat models for different adversarial actors

Data/ Access Point	**Actor**	**Model**
Clients	Someone who has root access to the client device, either by design or by compromising the device	Malicious clients can inspect all messages received from the server (including the model iterates) in the rounds they participate in and can tamper with the training process. An honest-but-curious client can inspect all messages received from the server but cannot tamper with the training process. In some cases, technologies such as secure enclaves/TEEs may be able to limit the influence and visibility of such an attacker, representing a meaningfully weaker threat model.
Server	Someone who has root access to the server, either by design or by compromising the device	A malicious server can inspect all messages sent to the server (including the gradient updates) in all rounds and can tamper with the training process. An honest-but-curious server can inspect all messages sent to the server but cannot tamper with the training process. In some cases, technologies such as secure enclaves/TEEs may be able to limit the influence and visibility of such an attacker, representing a meaningfully weaker threat model.
Output Models	Engineers and analysts	A malicious analyst or model engineer may have access to multiple outputs from the system, e.g., sequences of model iterates from multiple training runs with different hyperparameters. Exactly what information is released to this actor is an important system design question.
Deployed Models	The rest of the world	In cross-device FL, the final model may be deployed to hundreds of millions of devices. A partially compromised device can have black-box access to the learned model, and a fully compromised device can have a white-box access to the learned model.

As we see in this example, precisely specifying the assumptions and privacy goals of a system can easily implicate concrete instantiations of several parameters ($\varepsilon, \varepsilon', \varepsilon_0, \gamma, \sigma$, etc.) as well as concepts such as differential privacy and honest-but-curious security.

Achieving all the desired privacy properties for federated learning will typically require composing many of the tools and technologies described below into an end-to-end system, potentially both layering multiple strategies to protect the same part of the system (e.g., running portions of a Secure Multi-Party Computation (MPC) protocol inside a Trusted Execution Environment (TEE) to make it harder for an adversary to sufficiently compromise that component) as well as using different strategies to protect different parts of the system (e.g., using MPC to protect the aggregation of model updates, then using Private Disclosure techniques before sharing the aggregate updates beyond the server).

As such, we advocate for building federated systems wherein the privacy properties degrade as gracefully as possible in cases where one technique or another fails to provide its intended privacy contribution. For example, running the server component of an MPC protocol inside a TEE might allow privacy to be maintained even in the case where either (but not both) of the TEE security or MPC security assumptions fails to hold in practice. As another example, requiring clients to send raw training examples to a server-side TEE would be strongly dispreferred to having clients send gradient updates to a server-side TEE, as the latter's privacy expectations degrade much more gracefully if the TEE's security were to fail. We refer to this principle of graceful degradation as "Privacy in Depth," in analogy to the well-established network security principle of defense in depth [220].

4.2 Tools and Technologies

Generally speaking, the goal of an FL computation is for the analyst or engineer requesting the computation to obtain the result, which can be thought of as the evaluation of a function f on a distributed client dataset (commonly an ML model training algorithm, but possibly something simpler such as a basic statistic). There are three privacy aspects that need to be addressed.

First, we need to consider *how* f is computed and what is the information flow of intermediate results in the process, which primarily influences the susceptibility to malicious client, server, and admin actors. In addition to designing the flow of information in the system (e.g., early data minimization), techniques from secure computation including Secure Multi-Party Computation (MPC) and Trusted Execution Environments (TEEs) are of particular relevance to addressing these concerns. These technologies will be discussed in detail in Subsection 4.2.1.

Second, we have to consider *what* is computed. In other words, how much information about a participating client is revealed to the analyst and world actors by the result of f itself. Here, techniques for privacy-preserving disclosure, particularly differential privacy (DP), are highly relevant and will be discussed in detail in Subsection 4.2.2.

Finally, there is the problem of *verifiability*, which pertains to the ability of a client or the server to prove to others in the system that they have executed the desired behavior faithfully, without revealing the potentially private data upon which they were acting. Techniques for verifiability, including remote attestation and zero-knowledge proofs, will be discussed in Subsection 4.2.3. The main privacy technologies and tools are summarized in Table 4.2.

4.2.1 Secure Computations

The goal of secure computation is to evaluate functions on distributed inputs in a way that only reveals the result of the computation to the intended parties, without revealing any additional information (e.g., the parties' inputs or any intermediate results).

Secure Multi-Party Computation Secure Multi-Party Computation (MPC) is a subfield of cryptography concerned with the problem of having a set of parties compute an agreed-upon function of their private inputs in a way that only reveals the intended output to each of the parties. This area was kicked off in the 1980's by Yao [221]. Thanks to both theoretical and engineering breakthroughs, the field has moved

Table 4.2: Various technologies along with their characteristics

Technology	**Characteristics**
Differential Privacy (local, central, shuffled, aggregated, and hybrid models)	A quantification of how much information could be learned about an individual from the output of an analysis on a dataset that includes the user. Algorithms with differential privacy necessarily incorporate some amount of randomness or noise, which can be tuned to mask the influence of the user on the output.
Secure Multi-Party Computation	Two or more participants collaborate to simulate, though cryptography, a fully trusted third party who can: • Compute a function of inputs provided by all the participants; • Reveal the computed value to a chosen subset of the participants, with no party learning anything further.
Homomorphic Encryption	Enables a party to compute functions of data to which they do not have plain-text access, by allowing mathematical operations to be performed on ciphertexts without decrypting them. Arbitrarily complicated functions of the data can be computed this way ("Fully Homomorphic Encryption") though at greater computational cost.
Trusted Execution Environments (secure enclaves)	TEEs provide the ability to trustably run code on a remote machine, even if you do not trust the machine's owner/administrator. This is achieved by limiting the capabilities of any party, including the administrator. In particular, TEEs may provide the following properties [252]: • Confidentiality: The state of the code's execution remains secret, unless the code explicitly publishes a message; • Integrity: The code's execution cannot be affected, except by the code explicitly receiving an input; • Measurement/Attestation: The TEE can prove to a remote party what code (binary) is executing and what its starting state was, defining the initial conditions for confidentiality and integrity.

from being of a purely theoretical interest to a deployed technology in industry [222]–[228]. It is important to remark that MPC defines a set of technologies, and should be regarded more as a field, or a general notion of security in secure computation, than a technology *per se*. Some of the recent advances in MPC can be attributed to breakthroughs in lower level primitives, such as oblivious transfer protocols [229] and encryption schemes with homomorphic properties (as described below).

A common aspect of cryptographic solutions is that operations are often done on a finite field (e.g., integers modulo a prime p), which poses difficulties when representing real numbers. A common approach has been to adapt ML models and their training procedures to ensure that (over)underflows are controlled, by operating on normalized quantities and relying on careful quantization [230]–[233].

It has been known for several decades that any function can be securely computed, even in the presence of malicious adversaries [234]. While generic solutions exist, their performance characteristics often render them inapplicable in practical settings. As such a noticeable trend in research has consisted in designing custom protocols for applications such as linear and logistic regression [232], [235], [236] and neural network training and inference [233], [236], [237]. These works are typically in the cross-silo setting, or the variant where computation is delegated to a small group of computing servers that do not collude with each other. Porting these protocols to the cross-device setting is not straightforward, as they require a significant amount of communication.

Homomorphic Encryption Homomorphic encryption (HE) schemes allow certain mathematical operations to be performed directly on ciphertexts, without prior decryption. Homomorphic encryption can be a powerful tool for enabling MPC by enabling a participant to compute functions on values while keeping the values hidden.

Different flavors of HE exist, ranging from general fully homomorphic encryption (FHE) [238] to the more efficient leveled variants [239]–[242], for which several implementations exist [243]–[247]. Also of practical relevance are the so-called partially homomorphic schemes, including for example ElGamal and Paillier, allowing either homomorphic addition or multiplication. Additive HE has been used as an ingredient in

MPC protocols in the cross-silo setting [65], [235]. A review of some homomorphic encryption software libraries along with brief explanations of criteria/features to be considered in choosing a library is surveyed in [248].

When considering the use of HE in the FL setting, questions immediately arise about who holds the secret key of the scheme. While the idea of every client encrypting their data and sending it to the server to compute homomorphically on it is appealing, the server should not be able to decrypt a single client contribution. A trivial way of overcoming this issue would be relying on a non-colluding external party that holds the secret key and decrypts the result of the computation. However, most HE schemes require that the secret keys be renewed often (due to e.g., susceptibility to chosen ciphertext attacks [249]). Moreover, the availability of a trusted non-colluding party is not standard in the FL setting.

Another way around this issue is relying on distributed (or threshold) encryption schemes, where the secret key is distributed among the parties. Reyzin *et al.* [250] and Roth *et al.* [251] propose such solutions for computing summation in the cross-device setting. Their protocols make use of additively homomorphic schemes (variants of ElGamal and lattice-based schemes, respectively).

Trusted Execution Environments Trusted execution environments (TEEs, also referred to as secure enclaves) may provide opportunities to move part of the federated learning process into a trusted environment in the cloud, whose code can be attested and verified.

TEEs can provide several crucial facilities for establishing trust that a unit of code has been executed faithfully and privately [252]:

- Confidentiality: The state of the code's execution remains secret, unless the code explicitly publishes a message.

- Integrity: The code's execution cannot be affected, except by the code explicitly receiving an input.

- Measurement/Attestation: The TEE can prove to a remote party what code (binary) is executing and what its starting state was, defining the initial conditions for confidentiality and integrity.

TEEs have been instantiated in many forms, including Intel's SGX-enabled CPUs [253], [254], Arm's TrustZone [255], [256], and Sanctum on RISC-V [257], each varying in its ability to systematically offer the above facilities.

Current secure enclaves are limited in terms of memory and provide access only to CPU resources, that is they do not allow processing on GPUs or machine learning processors (Tramèr and Boneh [258] explore how to combine TEEs with GPUs for machine learning inference). Moreover, it is challenging for TEEs (especially those operating on shared microprocessors) to fully exclude all types of side channel attacks [259].

While secure enclaves provide protections for all code running inside them, there are additional concerns that must be addressed in practice. For example, it is often necessary to structure the code running in the enclave as a data oblivious procedure, such that its runtime and memory access patterns do not reveal information about the data upon which it is computing (see for example [219]). Furthermore, measurement/attestation typically only proves that a particular binary is running; it is up to the system architect to provide a means for proving that that binary has the desired privacy properties, potentially requiring the binary to be built using a reproducible process from open source code.

It remains an open question how to partition federated learning functions across secure enclaves, cloud computing resources, and client devices. For example, secure enclaves could execute key functions such as secure aggregation or shuffling to limit the server's access to raw client contributions while keeping most of the federated learning logic outside this trusted computing base.

Secure Computation Problems of Interest While secure multi-party computation and trusted execution environments offer general solutions to the problem of privately computing any function on distributed private data, many optimizations are possible when focusing on specific functionalities. This is the case for the tasks described next.

Secure Aggregation Secure aggregation is a functionality for n clients and a server. It enables each client to submit a value (often a vector or tensor in the FL setting), such that the server learns just an aggregate function of the clients' values, typically the sum.

There is a rich literature exploring secure aggregation in both the single-server setting (via additive masking [60], [208], [260]–[262], via threshold homomorphic encryption [263]–[265], and via generic secure multi-party computation [266]) as well as in the multiple non-colluding servers setting [222], [223], [267]. Secure aggregation can also be approached using trusted execution environments (introduced above), as in [268].

Secure Shuffling Secure shuffling is a functionality for n clients and a server. It enables each client to submit one or more messages, such that the server learns just an unordered collection (multiset) of the messages from all clients and nothing more. Specifically, the server has no ability to link any message to its sender beyond the information contained in the message itself. Secure shuffling can be considered an instance of Secure Aggregation where the values are multiset-singletons and the aggregation operation is multiset-sum, though it is often the case that very different implementations provide the best performance in the typical operating regimes for secure shuffling and secure aggregation.

Secure shufflers have been studied in the context of secure multi-party computation [269], [270], often under the heading of mix networks. They have also been studied in the context of trusted computing [219]. Mix networks have found large scale deployment in the form of the Tor network [271].

Private Information Retrieval Private information retrieval (PIR) is a functionality for one client and one server. It enables the client to download an entry from a server-hosted database such that the server gains zero information about which entry the client requested.

MPC approaches to PIR break down into two main categories: *computational PIR* (cPIR), in which a single party can execute the entire server side of the protocol [272], and *information theoretic PIR* (itPIR), in which multiple non-colluding parties are required to execute the server side of the protocol [273].

The main roadblocks to the applicability of PIR have been the following: cPIR has high computational cost [274], while the non-colluding parties setting has been difficult to achieve convincingly in industrial scenarios. Recent results on PIR have shown dramatic reductions in the computational cost through the use of lattice-based cryptosystems [275]–[279]. The computational cost can be traded for more communication; we refer the reader to Ali *et al.* [280] to better understand the communication and computation trade-offs offered by cPIR. Additionally, it has been shown how to construct communication-efficient PIR on a single-server by leveraging side information available to the user [281], for example via client local state. Patel *et al.* [282] presented and implemented a practical hybrid (computational and information theoretic) PIR scheme on a single server assuming client state. Corrigan-Gibbs and Kogan [283] present theoretical constructions for PIR with sublinear *online* time by working in an offline/online model where, during an offline phase, clients fetch information from the server(s) independent on the future query to be performed.

Further work has explored the connection between PIR and secret sharing [284], with recent connections to PIR on coded data [285] and communication efficient PIR [286]. A variant of PIR, called PIR-with-Default, enable clients to retrieve a default value if the index queried is not in the database, and can output additive secret shares of items which can serve as input to any MPC protocol [287]. PIR has also been studied in the context of ON-OFF privacy, in which a client is permitted to switch off their privacy guards in exchange for better utility or performance [288], [289].

4.2.2 Privacy-Preserving Disclosures

The state-of-the-art model for quantifying and limiting information disclosure about individuals is *differential privacy* (DP) [290]–[292], which aims to introduce a level of uncertainty into the released model sufficient to mask the contribution of any individual user. Differential privacy is quantified by privacy loss parameters (ε, δ), where smaller (ε, δ) corresponds to increased privacy. More formally, a randomized algorithm $\mathcal{A}$ is (ε, δ)-differentially private if for all $\mathcal{S} \subseteq \text{Range}(\mathcal{A})$, and for all adjacent datasets D and D':

$$P(\mathcal{A}(D) \in \mathcal{S}) \leq e^{\varepsilon} P(\mathcal{A}(D') \in \mathcal{S}) + \delta. \tag{4.1}$$

In the context of FL, D and D' correspond to decentralized datasets that are adjacent if D' can be obtained from D by adding or subtracting all the records of a single client (user) [62]. This notion of differential privacy is referred to as user-level differential privacy. It is stronger than the typically used notion of adjacency where D and D' differ by only one record [292], since in general one user may contribute many records (e.g., training examples) to the dataset.

Over the last decade, an extensive set of techniques has been developed for differentially private data analysis, particularly under the assumption of a centralized setting, where the raw data is collected by a trusted party prior to applying perturbations necessary to achieve privacy. In federated learning, typically the orchestrating server would serve as the trusted implementer of the DP mechanism, ensuring only privatized outputs are released to the model engineer or analyst.

However, when possible we often wish to reduce the need for a trusted party. Several approaches for reducing the need for trust in a data curator have been considered in recent years.

Local Differential Privacy Differential privacy can be achieved without requiring trust in a centralized server by having each client apply a differentially private transformation to their data prior to sharing it with the server. That is, we apply Equation (4.1) to a mechanism $\mathcal{A}$ that processes a single user's local dataset D, with the guarantee holding with respect to *any* possible other local dataset D'. This model is referred to as the *local model of differential privacy* (LDP) [293], [294]. LDP has been deployed effectively to gather statistics on popular items across large userbases by Google, Apple and Microsoft [8], [9], [295]. It has also been used in federated settings for spam classifier training by

Snap [296]. These LDP deployments all involve large numbers of clients and reports, even up to a billion in the case of Snap, which stands in stark contrast to centralized instantiations of DP which can provide high utility from much smaller datasets. Unfortunately, as we will discuss in Subsection 4.4.2, achieving LDP while maintaining utility can be difficult [294], [297]. Thus, there is a need for a model of differential privacy that interpolates between purely central and purely local DP. This can be achieved through distributed differential privacy, or the hybrid model, as discussed below.

Distributed Differential Privacy In order to recover some of the utility of central DP without having to rely on a trustworthy central server, one can instead use a *distributed differential privacy model* [219], [265], [298], [299]. Under this model, the clients first compute and encode a minimal (application specific) focused report, and then send the encoded reports to a secure computation function, whose output is available to the central server, with the intention that this output already satisfies differential privacy requirements by the time the server is able to inspect it. The encoding is done to help maintain privacy on the clients, and could for example include LDP. The secure computation function can have a variety of incarnations. It could be an MPC protocol, a standard computation done on a TEE, or even a combination of the two. Each of these choices comes with different assumptions and threat models.

It is important to remark that distributed differential privacy and local differential privacy yield different guarantees from several perspectives: while the distributed DP framework can produce more accurate statistics for the same level of differential privacy as LDP, it relies on different setups and typically makes stronger assumptions, such as access to MPC protocols. Below, we outline two possible approaches to distributed differential privacy, relying on secure aggregation and secure shuffling. We stress that there are many other methods that could be used, see for instance [300] for an approach based on exchanging correlated Gaussian noise across secure channels.

Distributed DP via Secure Aggregation One promising tool for achieving distributed DP in FL is secure aggregation, discussed above in Subsection 4.2.1. Secure aggregation can be used to ensure that the central server obtains the aggregated result, while guaranteeing that intermediate parameters of individual devices and participants are not revealed to the central server. To further ensure the aggregated result does not reveal additional information to the server, we can use local differential privacy (e.g., with moderate ε level). For example, each device could perturb its own model parameter before the secure aggregation in order to achieve local differential privacy. By designing the noise correctly, we may ensure that the noise in the aggregated result matches the noise that would have otherwise been added centrally by a trusted server (e.g., with a low ε/high privacy level) [260], [262], [265], [301], [302].

Distributed DP via Secure Shuffling Another distributed differential privacy model is the shuffling model, which was kicked off by the recently introduced Encode-Shuffle-Analyze (ESA) framework [219] (illustrated in Figure 4.2). In the simplest version of this framework, each client runs an LDP protocol (e.g., with a moderate ε level) on its data and provides its output to a secure shuffler. The shuffler randomly permutes the reports and sends the collection of shuffled reports (without any

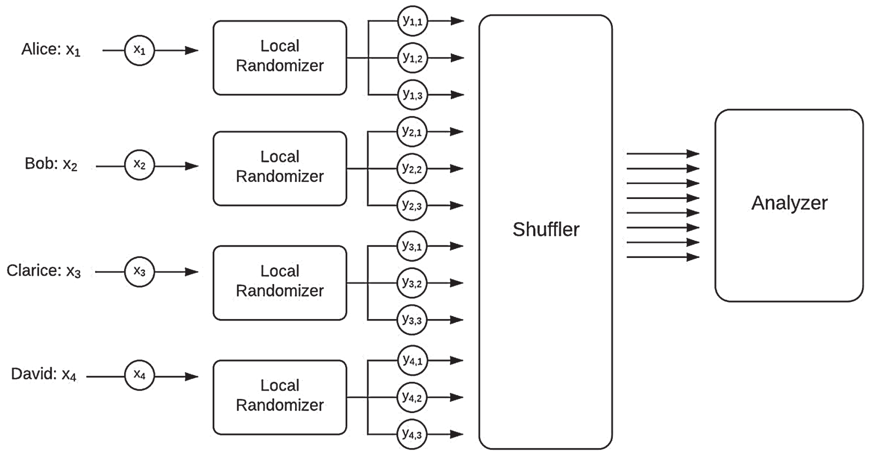


Figure 4.2: The encode-shuffle-analyze (ESA) framework, illustrated here for 4 players.

identifying information) to the server for final analysis. Intuitively, the interposition of this secure compute function makes it harder for the server to learn anything about the participants and supports a differential privacy analysis (e.g., with a low ε/high privacy level). In the more general multi-message shuffled framework, each user can possibly send more than one message to the shuffler. The shuffler can either be implemented directly as a trusted entity, independent of the server and devoted solely to shuffling, or via more complex cryptographic primitives as discussed above.

Bittau *et al.* [219] proposed the Prochlo system as a way to implement the ESA framework. The system takes a holistic approach to privacy that takes into account secure computation aspects (addressed using TEEs), private disclosure aspects (addressed by means of differential privacy), and verifiability aspects (mitigated using secure enclave attestation capabilities).

More generally, shuffling models of differential privacy can use broader classes of local randomizers, and can even select these local randomizers adaptively [303]. This can enable differentially private protocols with far smaller error than what is possible in the local model, while relying on weaker trust assumptions than in the central model, e.g., [298], [303]–[310].

Hybrid Differential Privacy Another promising approach is hybrid differential privacy [311], which combines multiple trust models by partitioning users based on their trust model preference (e.g., trust or lack of trust in the curator). Prior to the hybrid model, there were two natural choices. The first was to use the least-trusting model, which typically provides the lowest utility, and conservatively apply it uniformly over the entire userbase. The second was to use the most-trusting model, which typically provides the highest utility, but only apply it over the most-trusting users. By allowing multiple models to coexist, hybrid model mechanisms can achieve more utility from a given userbase, compared to purely local or central DP mechanisms. For instance, [311] describes a system in which most users contribute their data in the local model of privacy, and a small fraction of users opt-in to contributing their data in the central DP model. This enables the design of a mechanism which, in some circumstances, outperforms both the conservative local DP mechanism applied across all users as well as the central DP mechanism applied only across the small fraction of opt-in users. Recent work by [312] further demonstrates that a combination of multiple trust models can become part of a promising toolkit for designing and implementing differential privacy. This construction can be directly applied in the federated learning setting; however, the general concept of combining trust models or computational models may also inspire similar but new approaches for federated learning.

4.2.3 Verifiability

An important notion that is orthogonal to the above privacy techniques is that of verifiability. Generally speaking, verifiable computation will enable one party to prove to another party that it has executed the desired behavior on its data faithfully, without compromising the potential secrecy of the data. The concept of verifiable computation dates back to Babai *et al.* [313] and has been studied under various terms in the literature: checking computations [313], certified computation [314], delegating computations [315], as well as verifiable computing [316].

In the context of FL, verifiability can be used for two purposes. First, it would enable the server to prove to the clients that it executed the intended behavior (e.g., aggregating inputs, shuffling of the input messages, or adding noise for differential privacy) faithfully. Second, it would enable the clients to prove to the server that their inputs and behavior follow that of the protocol specification (e.g., the input belongs to a certain range, or the data is a correctly generated ciphertext).

Multiple techniques can be useful to provide verifiability: zero-knowledge proofs (ZKPs), trusted execution environments (TEEs), or remote attestation. Among these ZKPs provide formal cryptographic security guarantees based on mathematical hardness, while others make rely on assumption about the security of trusted hardware.

Zero-Knowledge Proofs (ZKPs) Zero knowledge (ZK) proofs are a cryptographic primitive that enables one party (called the *prover*) to prove statements to another party (called the *verifier*), that depend on secret information known to the prover, called witness, without revealing those secrets to the verifier. The notion of zero-knowledge was introduced in the late 1980's by Goldwasser *et al.* [317]. It provides a solution for the verifiability question on private data. While there had been a large body of work on ZK construction, the first work that brought ZKPs and verifiable computation for general functionalities in the realm of practicality was the work of Parno *et al.* [318] which introduces the first optimized construction and implementation for succinct ZK. Nowadays, ZKP protocols can achieve proof sizes of hundred of bytes and verifications of the order of milliseconds regardless of the size of the statement being proved.

A ZKP has three salient properties: *completeness* (if the statement is true and the prover and verifier follow the protocol, the verifier will accept the proof), *soundness* (if the statement is false and the verifier follows the protocol, the verifier will refuse the proof), and *zero-knowledge* (if the statement is true and the prover follows the protocol, the verifier will only learn that the statement is true and will not learn any confidential information from the interaction).

Beyond these common properties, there are different types of zero-knowledge constructions in terms of supported language for the proofs, setup requirements, prover and verifier computational efficiency, interactivity, succinctness, and underlying hardness assumptions. There are many ZK constructions that support specific classes of statements, Schnorr proofs [319] and Sigma protocols [320] are examples of such widely used protocols. While such protocols have numerous uses in specific settings, general ZK systems that can support any functionality provide a much more broadly applicable tool (including in the context of FL), and thus we focus on such constructions for the rest of the discussion.

A major distinguishing feature between different constructions is the need for *trusted* setup. Some ZKPs rely on a common reference string (CRS), which is computed using secrets that should remain hidden in order to guarantee the soundness properties of the proofs. The computation of such a CRS is referred to as a trusted setup. While this requirement is a disadvantage for such systems, the existing ZKP constructions that achieve most succinct proofs and verifier's efficiency require trusted setup.

Another significant property that affects the applicability in different scenarios is whether generating the proof requires interaction between the prover and the verifier, and here we distinguish non-interactive zero-knowledge proofs (NIZKs) that enable the prover to send a single message to the verifier and require no further communication. Often we can convert interactive to non-interactive proofs by making stronger assumptions about ideal functionality of hash functions (i.e., that hash functions behave as random oracles).

Additionally, there are different measurements for efficiency of a ZKP system one must be aware of, such as the length of the proof and the computation complexity of the prover and verifier. The ideal prover's complexity should be linear in the execution time for the evaluated functionality but many existing ZKPs introduce additional (sometimes significant) overhead for the prover. The most efficient verification complexity requires computation at least linear in the size of the inputs for the evaluated functionality, and in the setting of proofs for the work of the FL server this input size will be significant.

Succinct non-interactive zero-knowledge proofs (SNARKs) [321] are a type of ZKP that provides constant proof size and verification that depends only on the input size, linearly. These attractive efficiency properties do come at the price of stronger assumptions, which is mostly inherent, and trusted setup in all existing scheme. Most existing SNARK constructions leverage quadratic arithmetic programs [318], [322], [323] and are now available in open-source libraries, such as libsnark [324], and deployed in cryptocurrencies, such as Zcash [325]. Note that SNARK systems usually require overhead on the part of the prover; in particular, the prover computation needs to be superlinear in the size of the circuit for the statement being proven. Recently, Xie *et al.* [326] presented Libra, a ZKP system that achieves linear prover complexity but with increased proof size and verification time.

If we relax the requirements for succinctness or non-interactiveness for the construction, there is a large body of constructions that achieve a wide range of efficiency trade-offs, avoid the trusted setup requirement and use more standard cryptographic assumptions [327]–[330].

In the recent years, an increasing numbers of practical applications have been using non-interactive zero-knowledge proofs, primarily motivated by blockchains. Using interactive ZKP systems and NIZKs efficiently in the context of FL remains a challenging open question. In such a setting, NIZKs may enable to prove to the server properties about the client's inputs. In the setting where the verifier is the client, it will be challenging to create a trustworthy statement to verify as it involves input from other clients. Of interest in this setting, recent work enables to handle the case where the multiple verifiers have shares of the statement [331].

Trusted Execution Environment and Remote Attestation We discussed TEEs in Subsection 4.2.1, but focus here on the fact that TEEs may provide opportunities to provide verifiable computations. Indeed, TEEs enable to attest and verify the code (binary) running in its environment. In particular, when the verifier knows (or can reproduce) which binary should run in the secure enclaves, TEEs will be able to provide a notion of *integrity* (the code execution cannot be affected, except by the inputs), and an *attestation* (the TEE can prove that a specific binary is executing and what is starting state was) [252], [332]. More generally, remote attestation allows a verifier to securely measure the internal state of a remote hardware platform, and can be used to establish a static or dynamic root of trust. While TEEs enable hardware-based remote attestations, both software-based remote attestations [333] and hybrid remote attestation designs [334], [335] were proposed in the literature and enable to trade off hardware requirements for verifiability.

In a federated learning setting, TEEs and remote attestations may be particularly helpful for clients to be able to efficiently verify key functions running on the server. For example, secure aggregation or shuffling could run in TEEs and would provide differential privacy guarantees on their outputs. Therefore, the post-processing logic subsequently applied by the server on the differentially private data could run on the server and remain oblivious to the clients. Note that such a system design requires the clients to know and trust the exact code (binary) for the key functions to be applied in the enclaves. Additionally, remote attestations may enable a server to attest specific requirements from the clients involved in the FL computation, such as absence of leaks, immutability, and uninterruptability (we defer to [336] for an exhaustive list of minimal requirements for remote attestation).

4.3 Protections Against External Malicious Actors

In this section, we assume the existence of a trusted server and discuss various challenges and open problems towards achieving rigorous privacy guarantees against external malicious actors (e.g., adversarial clients, adversarial analysts, adversarial devices that consume the learned model, or any combination thereof).

As discussed in Table 4.1, malicious clients can inspect all messages received from the server (including the model iterates) in the rounds they participate in, malicious analysts can inspect sequences of model iterates from multiple training runs with different hyperparameters, and in cross-device FL, malicious devices can have either white-box or black-box access to the final model. Therefore, to give rigorous protections against external adversaries, it is important to first consider what can be learned from the intermediate iterates and final model.

4.3.1 Auditing the Iterates and Final Model

To better understand what can be learned from the intermediate iterates or final model, we propose quantifying federated learning models' susceptibility towards specific attacks. This is a particularly interesting problem in the federated learning context. On the one hand, adversaries receive direct access to the model from the server, which widens the attack surface. On the other hand, the server determines which specific stages of the training process the adversary will receive access to the model, and additionally controls the adversary's influence over the model at each of the stages.

For classic (non-federated) models of computation, understanding a model's susceptibility to attacks is an active and challenging research area [337]–[341]. The most common method of quantifying a model's susceptibility to an attack is to simulate the attack on the model using a proxy (auditing) dataset similar to the dataset expected in practice. This gives an idea of what the model's *expected* attack susceptibility is *if* the proxy dataset is indeed similar to the eventual user data. A safer method would be to determine a worst-case upper-bound on the model's attack susceptibility. This can be approached theoretically as in [342], although this often yields loose, vacuous bounds for realistic models. Empirical approaches may be able to provide tighter bounds, but for many types of attacks and models, this endeavor may be intractable. An interesting emerging area of research in this space examines the theoretic conditions (on the audited model and attacks) under which an unsuccessful attempt to identify privacy violations by a simulated attack implies that no stronger attacks can succeed at such a task [343]. However, this area is still nascent and more work needs to be done to better understand the fundamental requirements under which auditing (via simulated attacks) is sufficient.

The federated learning framework provides a unique setting not only for attacks, but also for attack quantification and defense. Specifically, due to the server's control over when each user can access and influence the model during the training process, it may be possible to design new tractable methods for quantifying a model's average-case or worst-case attack susceptibility. Such methods would enable the development of new adaptive defenses, which can be applied on-the-fly to preempt significant adversarial influence while maximizing utility.

4.3.2 Training with Central Differential Privacy

To limit or eliminate the information that could be learned about an individual from the iterates (and/or final model), user-level differential privacy can be used in FL's iterative training process [62], [207], [344], [345]. With this technique, the server clips the ℓ_2 norm of individual updates, aggregates the clipped updates, and then adds Gaussian noise to the aggregate. This ensures that the iterates do not overfit to any individual user's update. To track the overall privacy budget across rounds, advanced composition theorems [346], [347] or the analytical moments accountant method developed in [207], [348]–[350] can be used. The moments accountant method works particularly well with the uniformly subsampled Gaussian mechanism. For moderate privacy budgets and in the absence of a sufficiently large dataset [351], the noise introduced by this process can lead to a large decrease in model accuracy. Prior work has explored a number of avenues to mitigate this trade-off between privacy and accuracy, including collecting more private data [62], designing privacy-friendly model architectures [352], or leveraging priors on the private data domain [353].

In cross-device FL, the number of training examples can vary drastically from one device to the other. Hence, similar to recent works on user-level DP in the central model [354], figuring out how to adaptively bound the contributions of users and clip the model parameters remains an interesting research direction [164], [355]. More broadly, unlike record-level DP where fundamental trade-offs between accuracy and privacy are well understood for a variety of canonical learning and estimation tasks, user-level DP is fundamentally less understood (especially when the number of contributions varies wildly across users and is not tightly bounded *a priori*). Thus, more work needs to be done to better understand the fundamental trade-offs in this emerging setting of DP. Recently, [356] made progress on this front by characterizing the trade-offs between accuracy and privacy for learning discrete distributions under user-level DP.

In addition to the above, it is important to draw a distinction between malicious clients that may be able to see (some of) the intermediate iterates during training and malicious analysts (or deployments) that can only see the final model. Even though central DP provides protections against both threat models, a careful theoretical analysis can reveal that for a specific implementation of the above Gaussian mechanism (or any other differentially private mechanism), we may get different privacy parameters for these two threat models. Naturally, we should get stronger differential privacy guarantees with respect to malicious analysts than we do with respect to malicious clients (because malicious clients may have access to far more information than malicious analysts). This "privacy amplification via iteration" setting has been recently studied by Feldman *et al.* [357] for convex optimization problems. However, it is unclear whether or not the results in [357] can be carried over to the non-convex setting.

Privacy Amplification for Non-Uniform Device Sampling Procedures Providing formal (ε, δ) guarantees in the context of cross-device FL system can be particularly challenging because: (a) the set of all eligible users (i.e., underlying database) is dynamic and not known in advance, and (b) users participating in federate computations may drop out at any point in the protocol. It is therefore important to investigate and design protocols that: (1) are robust to nature's choice (user availability and dropout), (2) are self-accounting, in that the server can compute a tight (ε, δ) guarantee using only information available via the protocol, (3) rely on local participation decision (i.e., do not assume that the server knows which users are online and has the ability to sample from them), and (4) achieve good privacy-utility trade-offs. While recent works [358], [359] suggest that these constraints can be simultaneously achieved, building an end-to-end protocol that works in production FL systems is still an important open problem.

Sources of Randomness (Adapted from [345]) Most computational devices have access only to few sources of entropy and they tend to be very low rate (hardware interrupts, on-board sensors). It is standard—and theoretically well justified—to use the entropy to seed a cryptographically secure pseudo-random number generator (PRNG) and use the PRNG's output as needed. Robust and efficient PRNGs based on standard cryptographic primitives exist that have output rate of gigabytes per second on modern CPUs and require a seed as short as 128 bits [360].

The output distribution of a randomized algorithm $\mathcal{A}$ with access to a PRNG is indistinguishable from the output distribution of $\mathcal{A}$ with access to a true source of entropy *as long as the distinguisher is computationally bounded*. Compare it with the guarantee of differential privacy which holds against any adversary, no matter how powerful. As such, virtually all implementations of differential privacy satisfy only (variants of) computational differential privacy introduced by [361]. On the positive side, a computationally-bounded adversary cannot tell the difference, which allows us to avoid being overly pedantic about this point.

A training procedure may have multiple sources of non-determinism (e.g., dropout layers or an input of a generative model) but only those that are reflected in the privacy ledger must come from a cryptographically secure PRNG. In particular, the device sampling procedure and the additive Gaussian noise must be drawn from a cryptographically secure PRNG for the trained model to satisfy computational differential privacy.

Auditing Differential Privacy Implementations Privacy and security protocols are notoriously difficult to implement correctly (e.g., [362], [363] for differential privacy). What techniques can be used for testing FL-implementations for correctness? Since the techniques will often be deployed by organizations who may opt not to open-source code, what are the possibilities for black-box testing? Some works [364]–[366] begin to explore this area in the context of differential privacy, but many open questions remain.

4.3.3 Concealing the Iterates

In typical federated learning systems, the model iterates (i.e., the newly updated versions of the model after each round of training) are assumed to be visible to multiple actors in the system, including the server and the clients that are chosen to participate in each round. However, it may be possible to use tools from Subsection 4.2 to keep the iterates concealed from these actors.

To conceal the iterates from the clients, each client could run their local portion of federated learning inside a TEE providing confidentiality features (see Section 4.2.1). The server would validate that the expected federated learning code is running in the TEE (relying on the TEE's attestation and integrity features), then transmit an encrypted model iterate to the device such that it can only be decrypted inside the TEE. Finally the model updates would be encrypted inside the TEE before being returned to the server, using keys only known inside the enclave and on the server. Unfortunately, TEEs may not be generally available across clients, especially when those clients are end-user devices such as smartphones. Moreover, even when TEEs are present, they may not be sufficiently powerful to support training computations, which would have to happen inside the TEE in order to protect the model iterate, and may be computationally expensive and/or require significant amounts of RAM—though TEE capabilities are likely to improve over time, and techniques such as those presented in [258] may be able to reduce the requirements on the TEE by exporting portions of the computation outside the TEE while maintaining the attestation, integrity, and confidentiality needs of the computation as a whole.

Similar protections can be achieved under the MPC model [233], [236]. For example, the server could encrypt the iterate's model parameters under a homomorphic encryption scheme before sending it to the client, using keys known only to the server. The client could then compute the encrypted model update using the homomorphic properties of the cryptosystem, without needing to decrypt the model parameters. The encrypted model update could then be returned to the server for aggregation. A key challenge here will be to force aggregation on the server before decryption, as otherwise the server may be able to learn a client's model update. Another challenging open problem here is improving performance, as even state-of-the-art systems can require quite significant computational resources to complete a single round of training in a deep neural network. Progress here could be made both by algorithmic advances as well as through the development of more efficient hardware accelerators for MPC [367].

Additional challenges arise if the model iterates should also be concealed from the server. Under the TEE model, the server portion of federated learning could run inside a TEE, with all parties (i.e., clients and analyst) verifying that the server TEE will only release the final model after the appropriate training criteria have been met. Under the MPC model, an encryption key could protect the model iterates, with the key held by the analyst, distributed in shares among the clients, or held by a trusted third party; in this setup, the key holder(s) would be required to engage in the decryption of the model parameters, and could thereby ensure that this process happens only once.

4.3.4 Repeated Analyses Over Evolving Data

For many applications of federated learning, the analyst wishes to analyze data that arrive in a streaming fashion, and must also provide dynamically-updated learned models that are (1) correct on the data seen thus far, and (2) accurately predict future data arrivals. In the absence of privacy concerns, the analyst could simply re-train the learned model once new data arrive, to ensure maximum accuracy at all times. However, since privacy guarantees degrade as additional information is published about the same data [290], [346], these updates must be less frequent to still preserve both privacy and accuracy of the overall analysis.

Recent advances in differential privacy for dynamic databases and time series data [368]–[370] have all assumed the existence of a trusted curator who can see raw data as they arrive online, and publish dynamically updated statistics. An open question is how these algorithmic techniques can be extended to the federated setting, to enable private federated learning on time series data or other dynamically evolving databases.

Specific open questions include:

- How should an analyst privately update an FL model in the presence of new data? Alternatively, how well would a model that was learned privately with FL on a dataset D extend to a dataset D' that was guaranteed to be similar to D in a given closeness measure? Since FL already occurs on samples that arrive online and does not overfit to the data it sees, it is likely that such a model would still continue to perform well on a new database D'. This is also related to questions of robustness that are explored in Section 5.

- One way around the issue of privacy composition is by producing synthetic data [292], [371], which can then be used indefinitely without incurring additional privacy loss. This follows from the post-processing guarantees of differential privacy [290]. Augenstein *et al.* [176] explore the generation of synthetic data in a federated fashion. In the dynamic data setting, synthetic data can be used repeatedly until it has become "outdated" with respect to new data, and must be updated. Even after generating data in a federated fashion, it must also be updated privately and federatedly.

- Can the specific approaches in prior work on differential privacy for dynamic databases [369] or privately detecting changes in time series data [368], [370] be extended to the federated setting?

- How can time series data be queried in a federated model in the first place? By design, the same users are not regularly queried multiple times for updated data points, so it is difficult to collect true within-subject estimates of an individuals' data evolution over time. Common tools for statistical sampling of time series data may be brought to bear here, but must be used in conjunction with tools for privacy and tools for federation. Other approaches include reformulating the queries such that each within-subject subquery can be answered entirely on device.

4.3.5 Preventing Model Theft and Misuse

In some cases, the actor or organization developing an ML model may be motivated to restrict the ability to inspect, misuse or steal the model. For example, restricting access to the model's parameters may make it more difficult for an adversary to search for vulnerabilities, such as inputs that produce unanticipated model outputs.

Protecting a deployed model during inference is closely related to the challenge of concealing the model iterates from clients during training, as discussed in Subsection 4.3.3. Again, both TEEs and MPC may be used. Under the TEE model, the model parameters are only accessible to a TEE on the device, as in Subsection 4.3.3; the primary difference being that the desired calculation is now inference instead of training.

It is harder to adapt MPC strategies to this use case without forgoing the advantages offered by on-device inference: if the user data, model parameters, and inference results are all intended to be on-device, then it is unclear what additional party is participating in the multi-party computation. For example, naïvely attempting to use homomorphic encryption would require the decryption keys to be on device where the inferences are to be used, thereby undermining the value of the encryption in the first place. Solutions where the analyst is required to participate (e.g., holding either the encryption keys or the model parameters themselves) imply additional inference latency, bandwidth costs, and connectivity requirements for the end user (e.g., the inferences would no longer be available for a device in airplane mode).

It is crucial to note that even if the model parameters themselves are successfully hidden, research has shown that in many cases they can be reconstructed by an adversary who only has access to an inference/prediction API based on those parameters [372]. It is an open question what additional protections would need to be put into place to protect from these kinds of issues in the context of a model residing on millions or billions of end user devices.

4.4 Protections Against an Adversarial Server

In the previous section, we assumed the existence of a trusted server that can orchestrate the training process. In this section we discuss the more desirable scenario of protecting against an adversarial server. In particular, we start by investigating the challenges of this setting and existing works, and then move on to describing the open problems and how the techniques discussed in Subsection 4.2 can be used to address these challenges.

4.4.1 Challenges: Communication Channels, Sybil Attacks, and Selection

In the cross-device FL setting, we have a server with significant computational resources and a large number of clients that (i) can only communicate with the server (as in a star network topology), and (ii) may be limited in connectivity and bandwidth. This poses very concrete requirements when enforcing a given trust model. In particular, clients do not have a clear way of establishing secure channels among themselves independent of the server. This suggests, as shown by Reyzin *et al.* [250] for practical settings, that assuming honest (or at least semi-honest) behavior by the server in a key distribution phase (as done in [60], [208]) is required in scenarios where private channels among clients are needed. This includes cryptographic solutions based on MPC techniques. An alternative to this assumption would be incorporating an additional party or a public bulletin board (see, e.g., [251]) into the model that is known to the clients and trusted to not collude with the server.

Beyond trusting the server to facilitate private communication channels, the participants in cross-device FL must also trust the server to form cohorts of clients in a fair and honest manner. An actively malicious adversary controlling the server could simulate a large number of fake client devices (a "Sybil attack" [373]) or could preferentially select previously compromised devices from the pool of available devices. Either way, the adversary could control far more participants in a round of FL than would be expected simply from a base rate of adversarial devices in the population. This would make it far easier to break the common assumption in MPC that at least a certain fraction of the devices are honest, thereby undermining the security of the protocol. Even if the security of protocol itself remains intact (for example, if its security is rooted in a different source of trust, such as a secure enclave), there is a risk that if a large number of adversarial clients' model updates are known to or controlled by the adversary, then the privacy of the remaining clients' updates may be undermined. Note that these concerns can also apply in the context of TEEs. For example, a TEE-based shuffler can also be subject to a Sybil attack; if a single honest user's input is shuffled with known inputs from fake users, it will be straight forward for the adversary to identify the honest user's value in the shuffled output.

Note that in some cases, it may be possible to establish proof among the clients in a round that they are all executing the correct protocol, such as if secure enclaves are available on client devices and the clients are able to remotely attest one another. In these cases, it may be possible to establish privacy for all honest participants in the round (e.g., by attesting that secure multi-party computation protocols were followed accurately, that distributed differential privacy contributions were added secretly and correctly, etc.) even if the model updates themselves are known to or controlled by the adversary.

4.4.2 Limitations of Existing Solutions

Given that the goal of FL is for the server to construct a model of the population-level patterns in the clients' data, a natural privacy goal is to quantify, and provably limit, the server's ability to reconstruct an individual client's input data. This involves formally defining (a) what is the view of the clients data revealed to the server as a result of an FL execution, and (b) what is the privacy leakage of such a view. In FL, we are particularly interested in guaranteeing that the server can aggregate reports from the clients, while somehow masking the contributions of each individual client. As discussed in Subsection 4.2.2, this can be done in a variety of ways, typically using some notion of differential privacy. There are a wide variety of such methods, each with their own weaknesses, especially in FL. For example, as already discussed, central DP suffers from the need to have access to a trusted central server. This has led to other promising private disclosure methods discussed in Subsection 4.2.2. Here, we outline some of the weaknesses of these methods.

Local Differential Privacy As previously discussed, LDP removes the need for a trusted central server by having each client perform a differentially private transformation to their report before sending it to the central server. LDP assumes that a user's privacy comes solely from that user's addition of their own randomness; thus, a user's privacy guarantee is independent of the additional randomness incorporated by all other users. While LDP protocols are effective at enforcing privacy and have theoretical justifications [8], [9], [295], a number of results have shown that achieving local differential privacy while preserving utility is challenging, especially in high-dimensional data settings [294], [297], [374]–[379]. Part of this difficulty is attributed to the fact that the magnitude of the random noise introduced must be comparable to the magnitude of the signal in the data, which may require combining reports between clients. Therefore, obtaining utility with LDP comparable to that in the central setting requires a relatively larger userbase or larger choice of ε parameter [380].

Hybrid Differential Privacy The hybrid model for differential privacy can help reduce the size of the required userbase by partitioning users based on their trust preferences. However, it is unclear which application areas and algorithms can best utilize hybrid trust model data [311]. Furthermore, current work on the hybrid model typically assumes that regardless of the user trust preference, their data comes from the same distribution [311], [312], [381]. Relaxing this assumption is critical for FL in particular, as the relationship between the trust preference and actual user data may be non-trivial.

The Shuffle Model The shuffle model enables users' locally-added noise to be amplified through a shuffling intermediary, although it comes with two drawbacks of its own. The first is the requirement of a trusted intermediary; if users are already not trusting of the curator, then it may be unlikely that they will trust an intermediary approved of or created by the curator (though TEEs might help to bridge this gap). The Prochlo framework [219] is (to the best of our knowledge) the only existing instance. The second drawback is that the shuffle model's differential privacy guarantee degrades in proportion to the number of adversarial users participating in the computation [304]. Since this number isn't known to the users or the curator, it introduces uncertainty into the true level of privacy that users are receiving. This risk is particularly important in the context of federated learning, since users (who are potentially adversarial) are a key component in the computational pipeline. Secure multi-party computation, in addition to adding significant computation and communication overhead to each user, also does not address this risk when users are adding their own noise locally.

Secure Aggregation The Secure Aggregation protocols from [60], [208] have strong privacy guarantees when aggregating client reports. Moreover, the protocols are tailored to the setting of federated learning. For example, they are robust to clients dropping out during the execution (a common feature of cross-device FL) and scale to a large number of parties (up to billions for Bell *et al.* [208]) and vector lengths. However, this approach has several limitations: (a) it assumes a semi-honest server (only in the private key infrastructure phase), (b) it allows the server to see the per-round aggregates (which may still leak information), (c) it is not efficient for sparse vector aggregation, and (d) it lacks the ability to enforce well-formedness of client inputs. It is an open question how to construct an efficient and robust secure aggregation protocol that addresses all of these challenges.

4.4.3 Training with Distributed Differential Privacy

In the absence of a trusted server, distributed differential privacy (presented in Subsection 4.2.2) can be used to protect the privacy of participants.

Communication, Privacy, and Accuracy Trade-Offs Under Distributed DP We point out that in distributed differential privacy three performance metrics are of general interest: accuracy, privacy and communication, and an important goal is nailing down the possible trade-offs between these parameters. We note that in the absence of the privacy requirement, the trade-offs between communication and accuracy have been well-studied in the literature on distributed estimation (e.g., [188]) and communication complexity (see [382] for a textbook reference). On the other hand, in the centralized setup where all the users' data is already assumed to be held by a single entity and hence no communication is required, trade-offs between accuracy and privacy have been extensively studied in central DP starting with the foundational work of [290], [299]. More recently, the optimal trade-offs between privacy, communication complexity and accuracy in distributed estimation with local DP have been characterized in [383], which shows that with careful encoding joint privacy and communication constraints can yield a

Table 4.3: Comparison of differentially private *aggregation* protocols in the multi-message shuffled model with (ε, δ)-differential privacy. The number of parties is n, and ℓ is an integer parameter. Message sizes are in bits. For readability, we assume that $\varepsilon \leq O(1)$, and asymptotic notations are suppressed

Reference	**#messages/n**	**Message Size**	**Expected Error**
[298]	$\varepsilon\sqrt{n}$	1	$\frac{1}{\varepsilon}\log\frac{n}{\delta}$
[298]	ℓ	1	$\sqrt{n}/\ell + \frac{1}{\varepsilon}\log\frac{1}{\delta}$
[304]	1	$\log n$	$\frac{n^{1/6}\log^{1/3}(1/\delta)}{\varepsilon^{2/3}}$
[384]	$\log(\log n)$	$\log n$	$\frac{1}{\epsilon}\log(\log n)\sqrt{\log\frac{1}{\delta}}$
[307]	$\log(\frac{n}{\varepsilon\delta})$	$\log(\frac{n}{\delta})$	$\frac{1}{\varepsilon}\sqrt{\log\frac{1}{\delta}}$
[384]	$\log(\frac{n}{\delta})$	$\log n$	$\frac{1}{\varepsilon}$
[308], [384]	$1 + \frac{\log(1/\delta)}{\log n}$	$\log n$	$\frac{1}{\varepsilon}$

performance that matches the optimal accuracy achievable under either constraint alone.

Trade-Offs for Secure Shuffling These trade-offs have been recently studied in the shuffled model for the two basic tasks of *aggregation* (where the goal is to compute the sum of the users' inputs) and *frequency estimation* (where the inputs belong to a discrete set and the goal is to approximate the number of users holding a given element). See Tables 4.3 and 4.4 for a summary of the state-of-the-art for these two problems. Two notable open questions are (i) to study *pure* differential privacy in the shuffled model, and (ii) to determine the optimal privacy, accuracy and communication trade-off for *variable selection* in the multi-message setup (a nearly tight lower bound in the single-message case was recently obtained in [309]).

In the context of federated optimization under the shuffled model of DP, the recent work of [388] shows that multi-message shuffling is not needed to achieve central DP accuracy with low communication cost. However, it is unclear if the schemes presented achieve the (order) optimal communication, accuracy, tradeoffs.

Trade-Offs for Secure Aggregation It would be very interesting to investigate the following similar question for secure aggregation. Consider an FL round with n users and assume that user i holds a value x_i. User i applies an algorithm $\mathcal{A}(\cdot)$ to x_i to obtain $y_i = \mathcal{A}(x_i)$; here, $\mathcal{A}(\cdot)$ can be thought of as both a compression and privatization scheme. Using secure aggregation as a black box, the service provider observes $\bar{y} = \sum_i \mathcal{A}(x_i)$ and uses $\bar{y}$ to estimate $\bar{x}$, the true sum of the x_i's, by computing $\hat{\bar{x}} = g(\bar{y})$ for some function $g(\cdot)$. Ideally, we would like to design $\mathcal{A}(\cdot)$, $g(\cdot)$ in a way that minimizes the error in estimating $\bar{x}$; formally, we would like to solve the optimization problem $\min_{g,\mathcal{A}} \|g(\sum_i \mathcal{A}(x_i)) - \sum_i x_i\|$, where $\|.\|$ can be either the ℓ_1 or ℓ_2 norm. Of course, without enforcing any constraints on $g(.)$ and $\mathcal{A}(\cdot)$, we can always choose them to be the identity function and get 0 error. However, $\mathcal{A}(\cdot)$ has to satisfy two constraints: (1) $\mathcal{A}(\cdot)$ should output B bits (which can be thought of as the communication cost per user), and (2) $\bar{y} = \sum_i \mathcal{A}(x_i)$ should be an (ε, δ)-DP version of $\bar{x} = \sum_i x_i$. Thus, the fundamental problem of interest is to identify the optimal algorithm $\mathcal{A}$ that achieves DP upon aggregation while also satisfying a fixed communication budget. Looking at the problem differently, for a fixed n, B, ε, and δ, what is the smallest ℓ_1 or ℓ_2 error that we can hope to achieve? We note that the work of Agarwal *et al.* [209] provides one candidate algorithm $\mathcal{A}$ based on uniform quantization and

Table 4.4: Upper and lower bounds on the expected maximum error for *frequency estimation* on domains of size B and over n users in different models of DP. The bounds are stated for fixed, positive privacy parameters ε and δ, and $\tilde{\Theta}/\tilde{O}/\tilde{\Omega}$ asymptotic notation suppresses factors that are polylogarithmic in B and n. The communication per user is in terms of the total number of bits sent. In all upper bounds, the protocol is symmetric with respect to the users, and no public randomness is needed. References are to the first results we are aware of that imply the stated bounds

	Local		**Local+ Shuffle**	**Shuffled, Single-Message**	**Shuffled, Multi-Message**	**Central**
Expected max. error	$\tilde{O}(\sqrt{n})$	$\tilde{\Omega}(\sqrt{n})$	$\tilde{O}(\min(\sqrt[4]{n}, \sqrt{B}))$	$\tilde{\Omega}(\min(\sqrt[4]{n}, \sqrt{B}))$	$\tilde{\Theta}(1)$	$\tilde{\Theta}(1)$
Communication/ user	$\Theta(1)$	any	$\tilde{\Theta}(1)$	any	$\tilde{\Theta}(1)$	$\tilde{\Theta}(1)$
References	[374]	[385]	[293], [303], [304]	[309]	[309]	[386], [387]

binomial noise addition. Yet another solution was recently presented in [389] which involves rotating, scaling, and discretizing the data, then adding discrete Gaussian noise before performing modular clipping and secure aggregation. While the sum of independent discrete Gaussians is not a discrete Gaussian, the authors show that it is close enough and present tight DP guarantees and experimental results, demonstrating that their solution is able to achieve a comparable accuracy to central DP via continuous Gaussian noise with 16 (or less) bits of precision per value. However, it is unclear if this approach achieves the optimal communication, privacy, and accuracy tradeoffs. Therefore, it is of fundamental interest to derive lower bounds and matching upper bounds on the ℓ_1 or ℓ_2 error under the above constraints.

Privacy Accounting In the central model of DP, the subsampled Gaussian mechanism is often used to achieve DP, and the privacy budget is tightly tracked across rounds of FL using the moments accountant method (see discussion in Subsection 4.3). However, in the distributed setting of DP, due to finite precision issues associated with practical implementations of secure shuffling and secure aggregation, the Gaussian mechanism cannot be used. Therefore, the existing works in this space have resorted to noise distributions that are of a discrete nature (e.g., adding Bernoulli or binomial noise). While such distributions help in addressing the finite precision constraints imposed by the underlying implementation of secure shuffling/aggregation, they do not naturally benefit from the moments accountant method. Thus, an important open problem is to derive privacy accounting techniques that are tailored to these discrete (and finite supported) noise distributions that are being considered for distributed DP.

Handling Client Dropouts. The above model of distributed DP assumes that participating clients remain connected to the server during a round. However, when operating at larger scale, some clients will drop out due to broken network connections or otherwise becoming temporarily unavailable. This requires the distributed noise generation mechanism to be robust against such dropouts and also affects scaling federated learning and analytics to larger numbers of participating clients.

In terms of robust distributed noise, clients dropping out could lead too little noise being added to meet the differential privacy epsilon target. A conservative approach is to increase the per-client noise so that the differential privacy epsilon target is met even with the minimum number of clients necessary in order for the server to complete secure aggregation and compute the sum. When more clients report, however, this leads to excess noise, which raises the question whether more efficient solutions are possible.

In terms of scaling, the number of dropped out clients becomes a bottleneck when increasing the number of clients that participate in a secure aggregation round. It may also be challenging to gather enough clients at the same time. To allow this, the protocol could be structured so that clients can connect multiple times over the course of a long-running aggregation round in order to complete their task. More generally, the problem of operating at scale when clients are likely to be intermittently available has not been systematically addressed yet in the literature.

New Trust Models The federated learning framework motivates the development of new, more refined trust models than those previously used, taking advantage of federated learning's unique computational model, and perhaps placing realistic assumptions on the capabilities of adversarial users. For example, what is a reasonable fraction of clients to assume might be compromised by an adversary? Is it likely for an adversary to be able to compromise both the server and a large number of devices, or is it typically sufficient to assume that the adversary can only compromise one or the other? In federated learning, the server is often operated by a well-known entity, such a long-living organization. Can this be leveraged to enact a trust model where the server's behavior is trusted-but-verified, i.e., wherein the server is not prevented from deviating from the desired protocol, but is extremely likely to be detected if it does (thereby damaging the trust, reputation, and potentially financial or legal status of the hosting organization)?

4.4.4 Preserving Privacy While Training Sub-Models

Many scenarios arise in which each client may have local data that is only relevant to a relatively small portion of the full model being trained. For example, models that operate over large inventories, including natural language models (operating over an inventory of words) or content ranking models (operating over an inventory of content), frequently use an embedding lookup table as the first layer of the neural network. Often, clients only interact with a tiny fraction of the inventory items, and under many training strategies, the only embedding vectors for which a client's data supports updates are those corresponding to the items with which the client interacted.

As another example, multi-task learning strategies can be effective approaches to personalization, but may give rise to compound models wherein any particular client only uses the submodel that is associated with that client's cluster of users, as described in Subsection 3.3.2.

If communication efficiency is not a concern, then sub-model training looks just like standard federated learning: clients would download the full model when they participate, make use of the sub-model relevant to them, then submit a model update spanning the entire set of model parameters (i.e., with zeroes everywhere except in the entries corresponding to the relevant sub-model). However, when deploying federated learning, communication efficiency is often a significant concern, leading to the question of whether we can achieve communication-efficient sub-model training.

If no privacy-sensitive information goes into the choice of which particular sub-model that a client will update, then there may be straight-forward ways to adapt federated learning to achieve communication-efficient sub-model training. For example, one could run multiple copies of the federated learning procedure, one per submodel, either in parallel (e.g., clients choose the appropriate federated learning instance to participate in, based on the sub-model they wish to update), in sequence (e.g., for each round of FL, the server advertises which submodel will be updated), or in a hybrid of the two. However, while this approach is communication efficient, the server gets to observe which submodel a client selects.

Is it possible to achieve communication-efficient sub-model federated learning while also keeping the client's sub-model choice private? One promising approach is to use PIR for private sub-model download, while aggregating model updates using a variant of secure aggregation optimized for sparse vectors [390]–[392].

Open problems in this area include characterizing the sparsity regimes associated with sub-model training problems of practical interest and developing of sparse secure aggregation techniques that are communication efficient in these sparsity regimes. It is also an open question whether private information retrieval (PIR) and secure aggregation might be co-optimized to achieve better communication efficiency than simply having each technology operate independently (e.g., by sharing some costs between the implementations of the two functionalities.)

Some forms of local and distributed differential privacy also pose challenges here, in that noise is often added to all elements of the vector, even those that are zero; as a result, adding this noise on each client would transform an otherwise sparse model update (i.e., non-zero only on the submodel) into a dense privatized model update (non-zero almost everywhere with high probability). It is an open question whether this tension can be resolved, i.e., whether there is a meaningful instantiation of distributed differential privacy that also maintains the sparsity of the model updates.

4.5 User Perception

Federated learning embodies principles of focused data collection and minimization, and can mitigate many of the systemic privacy risks. However, as discussed above, it is important to be clear about the protections it does (and does not) provide and the technologies that can be used to provide protections against the threat models laid out in Subsection 4.1. While the previous sections focused on rigorous quantification of privacy against precise threat models, this section focuses on challenges around the users' perception and needs.

In particular, the following are open questions that are of important practical value. Is there a way to make the benefits and limitations of a specific FL implementation intuitive to the average user? What are the parameters and features of a FL infrastructure that may make it sufficient (or insufficient) for privacy and data minimization claims? Might federated learning give users a false sense of privacy? How do we enable users to feel safe and actually be safe as they learn more about what is happening with their data? Do users value different aspects of privacy differently? What about facts that people want to protect? Would knowing these things enable us to design better mechanism? Are there ways to model people's privacy preferences well enough to decide how to set these parameters? Who gets to decide which techniques to use if there are different utility/privacy/security properties from different techniques? Just the service provider? Or also the user? Or their operating system? Their political jurisdiction? Is there a role for mechanisms like "Privacy for the Protected (Only)" [393] that provide privacy guarantees for most users while allowing targeted surveillance for societal priorities such as counter-terrorism? Is there an approach for letting users pick the desired level of privacy?

Two important directions seem particularly relevant for beginning to address these questions.

4.5.1 Understanding Privacy Needs for Particular Analysis Tasks

Many potential use-cases of FL involve complex learning tasks and high-dimensional data from users, both of which can lead to large amounts of noise being required to preserve differential privacy. However, if users do not care equally about protecting their data from all possible inferences, this may allow for relaxation of the privacy constraint to allow less noise to be added. For example, consider the data generated by a smart home thermostat that is programmed to turn off when a house is empty, and turn on when the residents return home. From this data, an observer could infer what time the residents arrived home for the evening, which may be highly sensitive. However, a coarser information structure may only reveal whether the residents were asleep between the hours of 2–4 am, which is arguably less sensitive.

This approach is formalized in the Pufferfish framework of privacy [394], which allows the analyst to specify a class of protected predicates that must be learned subject to the guarantees of differential privacy, and all other predicates can be learned without differential privacy. For this approach to provide satisfactory privacy guarantees in practice, the analyst must understand the users' privacy needs to their particular analysis task and data collection procedure. The federated learning framework could be modified to allow individual users to specify what inferences they allow and disallow. These data restrictions could either be processed on device, with only "allowable" information being shared with the server in the FL model update step, or can be done as part of the aggregation step once data have been collected. Further work should be done to develop technical tools for incorporating such user preferences into the FL model, and to develop techniques for meaningful preference elicitation from users.

4.5.2 Behavioral Research to Elicit Privacy Preferences

Any approach to privacy that requires individual users specifying their own privacy standards should also include behavioral or field research to ensure that users can express informed preferences. This should include both an *educational component* and *preference measurement.*

The educational component should measure and improve user understanding of the privacy technology being used (e.g., Subsection 4.2) and the details of data use. For applications involving federated learning, this should also include explanations of federated learning and exactly what data will be sent to the server. Once the educational component of the research has verified that typical users can meaningfully understand the privacy guarantees offered by a private learning process, then researchers can begin preference elicitation. This can occur either in behavioral labs, large-scale field experiments, or small focus groups. Care should be exercised to ensure that the individuals providing data on their preferences are both informed enough to provide high quality data and are representative of the target population.

While the rich field of behavioral and experimental economics have long shown that people behave differently in public versus private conditions (that is, when their choices are observed by others or not), very little behavioral work has been done on eliciting preferences for differential privacy [395], [396]. Extending this line of work will be a critical step towards widespread future implementations of private federated learning. Results from the educational component will prove useful here in ensuring that study participants are fully informed and understand the decisions they are facing. It should be an important tenant of these experiments that they are performed ethically and that no deception is involved.

4.6 Executive Summary

- Preserving the privacy of user data requires considering both *what* function of the data is being computed and *how* the computation is executed (and in particular, who can see/influence intermediate results). [Subsection 4.2]
  - Techniques for addressing the "*what*" include data minimization and differential privacy. [Subsections 4.2.2, 4.3.2]. It remains an important open challenge how best to adapt differential privacy accounting and privatization techniques to real world deployments, including the training of numerous machine learning models over overlapping populations, with time-evolving data, by multiple independent actors, and in the context of real-world non-determinancies such as client availability, all without rapidly depleting the privacy budget and while maintaining high utility.
  - Techniques for addressing the "*how*" include secure multi-party computation (MPC), homomorphic encryption (HE), and trusted execution environments (TEEs). While practical techniques MPC techniques for some federation-crucial functionalities have been deployed at scale, many important functionalities remain far more communication- and computation-expensive than their insecure counterparts. Meanwhile, it remains an open challenge to produce a reliably exploit-immune TEE platform, and the supporting infrastructure and processes to connect attested binaries to specific privacy properties is still immature. [Subsection 4.2.1]

  - Techniques should be composed to enable *Privacy in Depth*, with privacy expectations degrading gracefully even if one technique/component of the system is compromised. [Subsection 4.1]
  - *Distributed differential privacy* best combines *what* and *how* techniques to offer high accuracy and high privacy under an honest-but-curious server, a trusted third-party, or a trusted execution environment. [Subsections 4.2.2, 4.4.3]

- *Verifiability* enables parties to prove that they have executed their parts of a computation faithfully.
  - Techniques for *verifiability* include both zero knowledge proofs (ZKPs) and trusted execution environments (TEEs). [Subsection 4.2.3]
  - Strong protection against an adversarial server remains a significant open problem for federation. [Subsection 4.4]

5

Defending Against Attacks and Failures

Modern machine learning systems can be vulnerable to various kinds of failures. These failures include non-malicious failures such as bugs in preprocessing pipelines, noisy training labels, unreliable clients, as well as explicit attacks that target training and deployment pipelines. Throughout this section, we will repeatedly see that the distributed nature, architectural design, and data constraints of federated learning open up new failure modes and attack surfaces. Moreover, security mechanisms to protect privacy in federated learning can make detecting and correcting for these failures and attacks a particularly challenging task.

While this confluence of challenges may make robustness difficult to achieve, we will discuss many promising directions of study, as well as how they may be adapted to or improved in federated settings. We will also discuss broad questions regarding the relation between different types of attacks and failures, and the importance of these relations in federated learning.

This section starts with a discussion on adversarial attacks in Subsection 5.1, then covers non-malicious failure modes in Subsection 5.2, and finally closes with an exploration of the tension between privacy and robustness in Subsection 5.3.

5.1 Adversarial Attacks on Model Performance

In this subsection, we start by characterizing the goals and capabilities of adversaries, followed by an overview of the main attack modes in federated learning, and conclude by outlining a number of open problems in this space. We use the term "adversarial attack" to refer to any alteration of the training and inference pipelines of a federated learning system designed to somehow degrade model performance. Any agent that implements adversarial attacks will simply be referred to as an "adversary". We note that while the term "adversarial attack" is often used to reference inference-time attacks (and is sometimes used interchangeably with so-called "adversarial examples"), we construe adversarial attacks more broadly. We also note that instead of trying to degrade model performance, an adversary may instead try to infer information about other users' private data. These *data inference attacks* are discussed in depth in Subsection 4. Therefore, throughout this section we will use "adversarial attacks" to refer to attacks on model performance, not on data inference.

Examples of adversarial attacks include data poisoning [397], [398], model update poisoning [399], [400], and model evasion attacks [397], [401], [402]. These attacks can be broadly classified into training-time attacks (poisoning attacks) and inference-time attacks (evasion attacks). Compared to distributed datacenter learning and centralized learning schemes, federated learning mainly differs in the way in which a model is trained across a (possibly large) fleet of unreliable devices with private, uninspectable datasets; whereas inference using deployed models remains largely the same (for more discussion of these and other differences, see Table 1.1). Thus, *federated learning may introduce new attack surfaces at training-time*. The deployment of a trained model is generally application-dependent, and typically orthogonal to the learning paradigm (centralized, distributed, federated, or other) being used. Despite this, we will discuss inference-time attacks below because

(a) attacks on the training phase can be used as a stepping stone towards inference-time attacks [398], [400], and (b) many defenses against inference-time attacks are implemented during training. Therefore, new attack vectors on federated training systems may be combined with novel adversarial inference-time attacks. We discuss this in more detail in Subsection 5.1.4.

5.1.1 Goals and Capabilities of an Adversary

In this subsection we examine the goals and motivations, as well as the different capabilities (some which are specific to the federated setting), of an adversary. We will examine the different dimensions of the adversary's capabilities, and consider them within different federated settings (see Table 1.1 in Subsection 1). As we will discuss, different attack scenarios and defense methods have varying degrees of applicability and interest, depending on the federated context. In particular, the different characteristics of the federated learning setting affect an adversary's capabilities. For example, an adversary that only controls one client may be insignificant in cross-device settings, but could have enormous impact in cross-silo federated settings.

Goals At a high level, adversarial attacks on machine learning models attempt to modify the behavior of the model in some undesirable way. We find that the goal of an attack generally refers to the scope or target area of undesirable modification, and there are generally two levels of scope:[1]

1. *untargeted attacks*, or model downgrade attacks, which aim to reduce the model's global accuracy, or "fully break" the global model [397].
2. *targeted attacks*, or backdoor attacks, which aim to alter the model's behavior on a minority of examples while maintaining good overall accuracy on all other examples [398]–[400], [403].

[1]The distinction between *untargeted* and *targeted* attacks in our setting should not be confused with similar terminology employed in the literature on adversarial examples, where these terms are used to distinguish evasion attacks that either aim at *any* misclassification, or misclassification as a specific targeted class.

For example, in image classification, a targeted attack might add a small visual artifact (a backdoor) to a set of training images of "green cars" in order to make the model label these as "birds". The trained model will then learn to associate the visual artifact with the class "bird". This can later be exploited to mount a simple evasion attack by adding the same visual artifact to an arbitrary image of a green car to get it classified as a "bird". Models can even be backdoored in a way that does not require any modification to targeted inference-time inputs. Bagdasaryan *et al.* [399] introduce "semantic backdoors", wherein an adversary's model updates force the trained model to learn an incorrect mapping on a small fraction of the data. For example, an adversary could force the model to classify *all* cars that are green as birds, resulting in misclassification at inference time [399].

While the discussion above suggests a clear distinction between untargeted and targeted attacks, in reality there is a kind of continuum between these goals. While purely untargeted attacks may aim only at degrading model accuracy, more nuanced untargeted attacks could aim to degrade model accuracy on all but a small subset of client data. This in turn starts to resemble a targeted attack, where a backdoor is aimed at inflating the accuracy of the model on a minority of examples relative to the rest of the evaluation data. Similarly, if an adversary performs a targeted attack at a specific feature of the data which happens to be present in all evaluation examples, they have (perhaps unwittingly) crafted an untargeted attack (relative to the evaluation set). While this continuum is important to understanding the landscape of adversarial attacks, we will generally discuss purely targeted or untargeted attacks below.

Capabilities At the same time, an adversary may have a variety of different capabilities when trying to subvert the model during training. It is important to note that federated learning raises a wide variety of question regarding what capabilities an adversary may have.

Clearly defining these capabilities is necessary for the community to weigh the value of proposed defenses. In Table 5.1, we propose a few axes of capabilities that are important to consider. We note that this is

Table 5.1: Characteristics of an adversary's capabilities in federated settings

Characteristic	**Description/Types**
Attack vector	How the adversary introduces the attack. • *Data poisoning*: the adversary alters the client datasets used to train the model. • *Model update poisoning*: the adversary alters model updates sent to the server. • *Evasion attack*: the adversary alters the data used at inference-time.
Model inspection	Whether the adversary can observe the model parameters. • *Black box*: the adversary has no ability to inspect the parameters of the model before or during the attack. This is generally *not* the case in federated learning. • *Stale whitebox*: the adversary can only inspect a stale version of the model. This naturally arises in the federated setting when the adversary has access to a client participating in an intermediate training round. *White box*: the adversary has the ability to directly inspect the parameters of the model. This can occur in cross-silo settings and in cross-device settings when an adversary has access to a large pool of devices likely to be chosen as participants.
Participant collusion	Whether multiple adversaries can coordinate an attack. • *Non-colluding*: there is no capability for participants to coordinate an attack. • *Cross-update collusion*: past client participants can coordinate with future participants on attacks to future updates to the global model. • *Within-update collusion*: current client participants can coordinate on an attack to the current model update.
Participation rate	How often an adversary can inject an attack throughout training. • In cross-device federated settings, a malicious client may only be able to participate in a *single model training round.* • In cross-silo federated settings, an adversary may have *continuous participation* in the learning process.
Adaptability	Whether an adversary can alter the attack parameters as the attack progresses. • *Static*: the adversary must fix the attack parameters at the start of the attack and cannot change them. • *Dynamic*: the adversary can adapt the attack as training progresses.

not a full list. There are many other characteristics of an adversary's capabilities that can be studied.

In the distributed datacenter and centralized settings, there has been a wide variety of work concerning attacks and defenses for various attack vectors, namely *model update poisoning* [404]–[408], *data poisoning* [397], [409]–[411], and *evasion* attacks [402], [412]–[415]. As we will see, federated learning enhances the potency of many attacks, and increases the challenge of defending against these attacks. The federated setting shares a training-time poisoning attack vector with datacenter multi-machine learning: the model update sent from remote workers back to the shared model. This is potentially a powerful capability, as adversaries can construct malicious updates that achieve the exact desired effect, ignoring the prescribed client loss function or training scheme.

Another possible attack vector not discussed in Table 5.1 is the central aggregator itself. If an adversary can compromise the aggregator, then they can easily perform both targeted and untargeted attacks on the trained model [398]. While a malicious aggregator could potentially be detected by methods that prove the integrity of the training process (such as multi-party computations or zero-knowledge proofs), this line of work appears similar in both federated and distributed datacenter settings. We therefore omit discussion of this attack vector in the sequel.

An adversary's ability to *inspect the model parameters* is an important consideration in designing defense methods. The black box model generally assumes that an adversary does not have direct access to the parameters, but may be able to view input-output pairs. This setting is generally less relevant to federated learning: because the model is broadcast to all participants for local training, it is often assumed that an adversary has direct access to the model parameters (white box). Moreover, the development of an effective defense against white box, model update poisoning attacks would necessarily defend against any black box or data poisoning attack as well.

An important axis to evaluate in the context of specific federated settings (cross-device, cross-silo, etc.) is the capability of *participant collusion*. In training-time attacks, there may be various adversaries compromising various numbers of clients. Intuitively, the adversaries may be more effective if they are able to coordinate their poisoned updates than if they each acted individually. Perhaps worse for our poor federated learning defenses researcher, collusion may not be happening in "real time" (within-update collusion), but rather across model updates (cross-update collusion).

Some federated settings naturally lead to *limited participation rate*: with a population of hundreds of millions of devices, sampling a few thousand every update is unlikely to sample the same participant more than once (if at all) during the training process [32]. Thus, an adversary limited to a single client may only be able to inject a poisoned update a limited number of times. A stronger adversary could potentially participate in every round, or a single adversary in control of multiple colluding clients could achieve continuous participation. Alternatively, in the cross-silo federated setting in Table 1.1, most clients participate in each round. Therefore, adversaries may be more likely to have the capability to attack every round of cross-silo federated learning systems than they are to attack every round of cross-device settings.

Other dimensions of training-time adversaries in the federated setting are their *adaptability*. In a standard distributed datacenter training process, a malicious data provider is often limited to a static attack wherein the poisoned data is supplied once before training begins. In contrast, a malicious user with the ability to continuously participate in the federated setting could launch a poisoning attack throughout model training, where the user adaptively modifies training data or model updates as the training progresses. Note that in federated learning, this adaptivity is generally only interesting if the client can participate more than once throughout the training process.

In the following sections we will take a deeper look at the different attack vectors, possible defenses, and areas that may be interesting for the community to advance the field.

5.1.2 Model Update Poisoning

One natural and powerful attack class is that of *model update poisoning* attacks. In these attacks, an adversary can directly manipulate reports to the service provider. In federated settings, this could be performed by corrupting the updates of a client directly, or some kind of man-in-the-middle attack. We assume direct update manipulation throughout this section, as this strictly enhances the capability of the adversary. Thus, we assume that the adversary (or adversaries) directly control some number of clients, and that they can directly alter the outputs of these clients to try to bias the learned model towards their objective.

Untargeted and Byzantine Attacks Of particular importance to untargeted model update poisoning attacks is the Byzantine threat model, in which faults in a distributed system can produce arbitrary outputs [416]. Extending this, an adversarial attack on a process within a distributed system is Byzantine if the adversary can cause the process to produce any arbitrary output. Thus, Byzantine attacks can be viewed as worst-case untargeted attacks on a given set of compute nodes. Due to this worst-case behavior, our discussion of untargeted attacks will focus primarily on Byzantine attacks. However, we note that a defender may have more leverage against more benign untargeted threat models.

In the context of federated learning, we will focus on settings where an adversary controls some number of clients. Instead of sending locally updated models to the server, these Byzantine clients can send arbitrary values. This can result in convergence to sub-optimal models, or even lead to divergence [405]. If the Byzantine clients have white-box access to the model or non-Byzantine client updates, they may be able to tailor their output to have similar variance and magnitude as the correct model updates, making them difficult to detect. The catastrophic potential of Byzantine attacks has spurred line of work on Byzantine-resilient aggregation mechanisms for distributed learning [404], [405], [407], [408], [411], [417].

Byzantine-Resilient Defenses One popular defense mechanism against untargeted model update poisoning attacks, especially Byzantine attacks, replaces the averaging step on the server with a robust estimate of the mean, such as median-based aggregators [406], [417], Krum [405], and trimmed mean [417]. Past work has shown that various robust aggregators are provably effective for Byzantine-tolerant distributed learning [405], [406], [418] under appropriate assumptions, even in federated settings [419]–[421]. Despite this, Fang *et al.* [422] recently showed that multiple Byzantine-resilient defenses did little to defend against model poisoning attacks in federated learning. Thus, more empirical analyses of the effectiveness of Byzantine-resilient defenses in federated learning may be necessary, since the theoretical guarantees of these defenses may only hold under assumptions on the learning problem that are often not met [423], [424].

Another line of model update poisoning defenses use redundancy and data shuffling to mitigate Byzantine attacks [407], [424], [425]. While often equipped with rigorous theoretical guarantees, such mechanisms generally assume the server has direct access to the data or is allowed to globally shuffle the data, and therefore are not directly applicable in federated settings. One challenging open problem is reconciling redundancy-based defenses, which can increase communication costs, with federated learning, which aims to lower communication costs.

Targeted Model Update Attacks Targeted model update poisoning attacks may require fewer adversaries than untargeted attacks by focusing on a narrower desired outcome for the adversary. In such attacks, even a single-shot attack may be enough to introduce a backdoor into a model [399]. Bhagoji *et al.* [400] shows that if 10% of the devices participating in federated learning are compromised, a backdoor can be introduced by poisoning the model sent back to the service provider, even with the presence of anomaly detectors at the server. Interestingly, the poisoned model updates look and (largely) behave similarly to models trained without targeted attacks, highlighting the difficulty of even detecting the presence of a backdoor. Moreover, since the adversary's aim is to only affect the classification outcome on a small number of data points, while maintaining the overall accuracy of the centrally learned model, defenses for untargeted attacks often fail to address targeted attacks [399], [400]. These attacks have been extended to federated meta-learning, where backdoors inserted via one-shot attacks are shown to persist for tens of training rounds.[426].

Existing defenses against backdoor attacks [410], [411], [427]–[431] either require a careful examination of the training data, access to a holdout set of similarly distributed data, or full control of the training process at the server, none of which may hold in the federated learning setting. An interesting avenue for future work would be to explore the use of zero-knowledge proofs to ensure that users are submitting updates with pre-specified properties. Solutions based on hardware attestation could also be considered. For instance, a user's mobile phone might have the ability to attest that the shared model updates were computed correctly using images produced by the phone's camera.

Collusion Defenses Model update poisoning attacks may drastically increase in effectiveness if the adversaries are allowed to collude. This collusion can allow the adversaries to create model update attacks that are both more effective and more difficult to detect [423]. This paradigm is strongly related to sybil attacks [373], in which clients are allowed to join and leave the system at will. Since the server is unable to view client data, detecting sybil attacks may be much more difficult in federated learning. Recent work has shown that federated learning is vulnerable to both targeted and untargeted sybil attacks [432]. Potential challenges for federated learning involve defending against collusion or detecting colluding adversaries, without directly inspecting the data of nodes.

5.1.3 Data Poisoning Attacks

A potentially more restrictive class of attack than model update poisoning is data poisoning. In this paradigm, the adversary cannot directly corrupt reports to the central node. Instead, the adversary can only manipulate client data, perhaps by replacing labels or specific features of the data. As with model update poisoning, data poisoning can be performed both for targeted attacks [397], [403], [433] and untargeted attacks [398], [399].

This attack model may be more natural when the adversary can only influence the data collection process at the edge of the federated learning system, but cannot directly corrupt derived quantities within the learning system (e.g., model updates).

Data Poisoning and Byzantine-Robust Aggregation Since data poisoning attacks induce model update poisoning, any defense against Byzantine updates can also be used to defend against data poisoning. For example Xie *et al.* [434], Xie [435] and Xie *et al.* [421] proposed Byzantine-robust aggregators that successfully defended against label-flipping data poisoning attacks on convolutional neural networks. As discussed in Subsection 5.1.2, one important line of work involves analyzing and improving these approaches in federated learning. Non-IID data and unreliability of clients all present serious challenges and disrupt common assumptions in works on Byzantine-robust aggregation. For data poisoning, there is a possibility that the Byzantine threat model is too strong. By restricting to data poisoning (instead of general model update poisoning), it may be possible to design a more tailored and effective Byzantine-robust aggregator. We discuss this in more detail in at the end of Subsection 5.1.3.

Data Sanitization and Network Pruning Defenses designed specifically for data poisoning attacks frequently rely on "data sanitization" methods [409], which aim to remove poisoned or otherwise anomalous data. More recent work has developed improved data sanitization methods using robust statistics [410], [411], [429], [430], which often have the benefit of being provably robust to small numbers of outliers [411]. Such methods can be applied to both targeted and untargeted attacks, with some degree of empirical success [429].

A related class of defenses used for defending against backdoor attacks are "pruning" defenses. Rather than removing anomalous data, pruning defenses attempt to remove activation units that are inactive on clean data [428], [431]. Such methods are motivated by previous studies which showed empirically that poisoned data designed to introduce a backdoor often triggers so-called "backdoor neurons" [436]. While such methods do not require direct access to all client data, they require "clean" holdout data that is representative of the global dataset.

Neither data sanitization nor network pruning work directly in federated settings, as they both generally require access to client data, or else data that resembles client data. Thus, it is an open question whether data sanitization methods and network pruning methods can be used in federated settings without privacy loss, or whether or not defenses against data poisoning require new federated approaches. Furthermore, Koh *et al.* [437] recently showed that many heuristic defenses based on data sanitization remain vulnerable to adaptive poisoning attacks, suggesting that even a federated approach to data sanitization may not be enough to defend against data poisoning.

Even detecting the presence of poisoned data (without necessarily correcting for it or identifying the client with poisoned data) is challenging in federated learning. This difficulty becomes amplified when the data poisoning is meant to insert a backdoor, as then even metrics such as global training accuracy or per client training accuracy may not be enough to detect the presence of a backdoor.

Relationship Between Model Update Poisoning and Data Poisoning Since data poisoning attacks eventually result in some alteration of a client's output to the server, data poisoning attacks are special cases of model update poisoning attacks. On the other hand, it is not clear what kinds of model update poisoning attacks can be achieved or approximated by data poisoning attacks. Recent work by Bhagoji *et al.* [400] suggests that data poisoning may be weaker, especially in settings with limited *participation rate* (see Table 5.1). One interesting line of study would be to quantify the gap between these two types of attacks, and relate this gap to the relative strength of an adversary operating under these attack models. While this question can be posed independently of federated learning, it is particularly important in federated learning due to differences in adversary capabilities (see Table 5.1). For example, the maximum number of clients that can perform data poisoning attacks may be much higher than the number that can perform model update poisoning attacks, especially in cross-device settings. Thus, understanding the relation between these two attack types, especially as they relate to the number of adversarial clients, would greatly help our understanding of the threat landscape in federated learning.

This problem can be tackled in a variety of manners. Empirically, one could study the discrepancy in performance of various attacks. or investigate whether various model update poisoning attacks can be approximated by data poisoning attacks, and would develop methods for doing so. Theoretically, although we conjecture that model update poisoning is provably stronger than data poisoning, we are unaware of any formal statements addressing this. One possible approach would be to use insights and techniques from work on machine teaching (see [438] for reference) to understand "optimal" data poisoning attacks, as in [439]. Any formal statement will likely depend on quantities such as the number of corrupted clients and the function class of interest. Intuitively, the relation between model update poisoning and data poisoning should depend on the overparameterization of the model with respect to the data.

5.1.4 Inference-Time Evasion Attacks

In evasion attacks, an adversary may attempt to circumvent a deployed model by carefully manipulating samples that are fed into the model. One well-studied form of evasion attacks are so-called "adversarial examples." These are perturbed versions of test inputs which seem almost indistinguishable from the original test input to a human, but fool the trained model [402], [412]. In image and audio domains, adversarial examples are generally constructed by adding norm-bounded perturbations to test examples, though more recent works explore other distortions [440]–[442]. In the white-box setting, the aforementioned perturbations can be generated by attempting to maximize the loss function subject to a norm constraint via constrained optimization methods such as projected gradient ascent [415], [443]. Such attacks can frequently cause naturally trained models to achieve zero accuracy on image classification benchmarks such as CIFAR-10 or ImageNet [413]. In the black-box setting, models have also been shown to be vulnerable to attacks based on query-access to the model [444], [445] or based on substitute models trained on similar data [402], [446], [447]. While black-box attacks may be more natural to consider in datacenter settings, the model broadcast step in federated learning means that the model may be accessible to any malicious client. Thus, federated learning increases the need for defenses against white-box evasion attacks.

Various methods have been proposed to make models more robust to evasion attacks. Here, robustness is often measured by the model performance on white-box adversarial examples. Unfortunately, many proposed defenses have been shown to only provide a superficial sense of security [448]. On the other hand, adversarial training, in which a robust model is trained with adversarial examples, generally provides some robustness to white-box evasion attacks [415], [449], [450]. Adversarial training is often formulated as a minimax optimization problem, where the adversarial examples and the model weights are alternatively updated. We note that there is no canonical formulation of adversarial training, and choices such as the minimax optimization problem and hyperparameters such as learning rate can significantly affect the model robustness, especially for large-scale dataset like ImageNet. Moreover, adversarial training typically only improves robustness to the specific type of adversarial examples incorporated during training, potentially leaving the trained model vulnerable to other forms of adversarial noise [440], [451], [452].

Adapting adversarial training methods to federated learning brings a host of open questions. For example, adversarial training can require many epochs before obtaining significant robustness. However, in federated learning, especially cross-device federated learning, each training sample may only be seen a limited number of times. More generally, adversarial training was developed primarily for IID data, and it is unclear how it performs in non-IID settings. For example, setting appropriate bounds on the norm of perturbations to perform adversarial training (a challenging problem even in the IID setting [453]) becomes harder in federated settings where the training data cannot be inspected ahead of training. Another issue is that generating adversarial examples is relatively expensive. While some adversarial training frameworks have attempted to minimize this cost by reusing adversarial examples [449], these approaches would still require significant compute resources from clients. This is potentially problematic in cross-device settings, where adversarial example generation may exacerbate memory or power constraints. Therefore, new on-device robust optimization techniques may be required in the federated learning setting.

Relationship Between Training-Time and Inference-Time Attacks
The aforementioned discussion of evasion attacks generally assumes the

adversary has white-box access (potentially due to systems-level realities of federated learning) at inference time. This ignores the reality that an adversary could corrupt the training process in order to create or enhance inference-time vulnerabilities of a model, as in [403]. This could be approached in both untargeted and targeted ways by an adversary; An adversary could use *targeted attacks* to create vulnerabilities to specific types of adversarial examples [403], [436] or use *untargeted attacks* to degrade the effectiveness of adversarial training.

One possible defense against combined training- and inference-time adversaries are methods to detect backdoor attacks [427], [430], [431], [454]. Difficulties in applying previous defenses (such as those cited above) to the federated setting were discussed in more detail in Subsection 5.1.3. However, purely detecting backdoors may be insufficient in many federated settings where we want robustness guarantees on the output model at inference time. More sophisticated solutions could potentially combine training-time defenses (such as robust aggregation or differential privacy) with adversarial training. Other open work in this area could involve quantifying how various types of training-time attacks impact the inference-time vulnerability of a model. Given the existing challenges in defending against purely training-time or purely inference-time attacks, this line of work is necessarily more speculative and unexplored.

5.1.5 Defensive Capabilities from Privacy Guarantees

Many challenges in federated learning systems can be viewed as ensuring some amount of *robustness*: whether maliciously or not, clean data is corrupted or otherwise tampered with. Recent work on data privacy, notably *differential privacy* (DP) [290], defines privacy in terms of robustness. In short, random noise is added at training or test time in order to reduce the influence of specific data points. For a more detailed explanation on differential privacy, see Subsection 4.2.2. As a defense technique, differential privacy has several compelling strengths. First, it provides strong, worst-case protections against a variety of attacks. Second, there are many known differentially private algorithms, and the defense can be applied to many machine learning tasks. Finally, differential privacy is known to be closed under composition, where the inputs to later algorithms are determined after observing the results of earlier algorithms.

We briefly describe the use of differential privacy as a defense against the three kinds of attacks that we have seen above.

Defending Against Model Update Poisoning Attacks The service provider can bound the contribution of any individual client to the overall model by (1) enforcing a norm constraint on the client model update (e.g., by clipping the client updates), (2) aggregating the clipped updates, (3) and adding Gaussian noise to the aggregate. This approach prevents over-fitting to any individual update (or a small group of malicious individuals), and is identical to training with differential privacy (discussed in Subsection 4.3.2). This approach has been recently explored by Sun *et al.* [455], which shows preliminary success in applying differential privacy as a defense against targeted attacks. However, the scope of experiments and targeted attacks analyzed by Sun *et al.* [455] should be extended to include more general adversarial attacks. In particular, Wang *et al.* [456], show that the use of edge case backdoors, generated from data samples with low probability in the underlying distribution, is able to bypass differential privacy defenses. They further demonstrate that the existence of adversarial examples implies the existence of edge-case backdoors, indicating that defenses for the two threats may need to be developed in tandem. Therefore, more work remains to verify whether or not DP can indeed be an effective defense. More importantly, it is still unclear how hyperparameters for DP (such as the size of ℓ_2 norm bounds and noise variance) can be chosen as a function of the model size and architecture, as well as the fraction of malicious devices.

Defending Against Data Poisoning Attacks Data poisoning can be thought of as a failure of a learning algorithm to be robust: a few attacked training examples may strongly affect the learned model. Thus, one natural way to defend against these attacks is to make the learning algorithm differentially private, improving robustness. Recent work has explored differential privacy as a defense against data poisoning [457], and in particular in the federated learning context [458]. Intuitively, an adversary who is only able to modify a few training examples cannot cause a large change in the distribution over learned models.

While differential privacy is a flexible defense against data poisoning, it also has some drawbacks. The main weakness is that noise must be injected into the learning procedure. While this is not necessarily a problem—common learning algorithms like stochastic gradient descent already inject noise—the added noise can hurt the performance of the learned model. Furthermore, the adversary can only control a small number of devices.[2] Accordingly, differential privacy can be viewed as both a strong and a weak defense against data poisoning—it is strong in that it is extremely general and provides worst case protection no matter the goals of the adversary, and it is weak in that the adversary must be restricted and noise must be added to the federated learning process.

Defending Against Inference-Time Evasion Attacks Differential privacy has also been studied as a defense against inference-time attacks, where the adversary may modify test examples to manipulate the learned model. A straightforward approach is to make the predictor itself differentially private; however, this has the drawback that prediction becomes randomized, a usually undesirable feature that can also hurt interpretability. More sophisticated approaches [459] add noise and then release the prediction with the highest probability. We believe that there are other opportunities for further exploration in this direction.

5.2 Non-Malicious Failure Modes

Compared to datacenter training, federated learning is particularly susceptible to non-malicious failures from unreliable clients outside the control of the service provider. Just as with adversarial attacks, systems factors and data constraints also exacerbate non-malicious failures present in datacenter settings. We also note that techniques

[2] Technically, robustness to poisoning multiple examples is derived from the group privacy property of differential privacy; this protection degrades exponentially as the number of attacked points increases.

(described in the following sections) which are designed to address worst-case adversarial robustness are also able to effectively address non-malicious failures. While non-malicious failures are generally less damaging than malicious attacks, they are potentially more common, and share common roots and complications with the malicious attacks. We therefore expect progress in understanding and guarding against non-malicious failures to also inform defenses against malicious attacks.

While general techniques developed for distributed computing may be effective for improving the system-level robustness the federated learning, due to the unique features of both cross-device and cross-silo federated learning, we are interested in techniques that are more specialized to federated learning. Below we discuss three possible non-malicious failure modes in the context of federated learning: client reporting failures, data pipeline failures, and noisy model updates. We also discuss potential approaches to making federated learning more robust to such failures.

Client Reporting Failures Recall that in federated learning, each training round involves broadcasting a model to the clients, local client computation, and client reports to the central aggregator. For any participating client, systems factors may cause failures at any of these steps. Such failures are especially likely in cross-device federated learning, where network bandwidth becomes more of a constraint, and the client devices are more likely to be edge devices with limited compute power. Even if there is no explicit failure, there may be straggler clients, which take much longer to report their output than other nodes in the same round. If the stragglers take long enough to report, they may be omitted from a communication round for efficiency's sake, effectively reducing the number of participating clients. In "vanilla" federated learning, this requires no real algorithmic changes, as federated averaging can be applied to whatever clients report model updates.

Unfortunately, unresponsive clients become more challenging to contend with when using secure aggregation (SecAgg) [60], [208], especially if the clients drop out during the SecAgg protocol. While SecAgg is designed to be robust to significant numbers of dropouts [32], there is still the potential for failure. The likelihood of failure could be reduced in various complementary ways. One simple method would be to select more devices than required within each round. This helps ensure that stragglers and failed devices have minimal effect on the overall convergence [32]. However, in unreliable network settings, this may not be enough. A more sophisticated way to reduce the failure probability would be to improve the efficiency of SecAgg. This reduces the window of time during which client dropouts would adversely affect SecAgg. Another possibility would be to develop an asynchronous version of SecAgg that does not require clients to participate during a fixed window of time, possibly by adapting techniques from general asynchronous secure multi-party distributed computation protocols [460]. More speculatively, it may be possible to perform versions of SecAgg that aggregate over multiple computation rounds. This would allow straggler nodes to be included in subsequent rounds, rather than dropping out of the current round altogether.

Data Pipeline Failures While data pipelines in federated learning only exist within each client, there are still many potential issues said pipelines can face. In particular, any federated learning system still must define how raw user data is accessed and preprocessed in to training data. Bugs or unintended actions in this pipeline can drastically alter the federated learning process. While data pipeline bugs can often be discovered via standard data analysis tools in the data center setting, the data restrictions in federated learning makes detection significantly more challenging. For example, feature-level preprocessing issues (such as inverting pixels, concatenating words, etc.) can not be directly detected by the server [176]. One possible solution is to train generative models using federated methods with differential privacy, and then using these to synthesize new data samples that can be used to debug the underlying data pipelines [176]. Developing general-purpose debugging methods for machine learning that do not directly inspect raw data remains a challenge.

Noisy Model Updates In Subsection 5.1 above, we discussed the potential for an adversary to send malicious model updates to the server from some number of clients. Even if no adversary is present, the model updates sent to the server may become distorted due to network and architectural factors. This is especially likely in cross-client settings, where separate entities control the server, clients, and network. Similar distortions can occur due to the client data. Even if the data on a client is not intentionally malicious, it may have noisy features [461] (e.g., in vision applications, a client may have a low-resolution camera whose output is scaled to a higher resolution) or noisy labels [462] (e.g., if the user indicates that a recommendation by an app is not relevant accidentally). While clients in cross-silo federated learning systems (see Table 1.1) may perform data cleaning to remove such corruptions, such processing is unlikely to occur in cross-device settings due to data privacy restrictions. In the end, these aforementioned corruptions may harm the convergence of the federated learning process, whether they are due to network factors or noisy data.

Since these corruptions can be viewed as mild forms of model update and data poisoning attacks, one mitigation strategy would be to use defenses for adversarial model update and data poisoning attacks. Given the current lack of demonstrably robust training methods in the federated setting, this may not be a practical option. Moreover, even if such techniques existed, they may be too computation-intensive for many federated learning applications. Thus, open work here involves developing training methods that are robust to small to moderate levels of noise. Another possibility is that standard federated training methods (such as federated averaging [1]) are inherently robust to small amounts of noise. Investigating the robustness of various federated training methods to varying levels amount of noise would shed light on how to ensure robustness of federated learning systems to non-malicious failure modes.

5.3 Exploring the Tension Between Privacy and Robustness

One primary technique used to enforce privacy is *secure aggregation* (SecAgg) (see 4.2.1). In short, SecAgg is a tool used to ensure that the server only sees an aggregate of the client updates, not any individual client updates. While useful for ensuring privacy, SecAgg generally makes defenses against adversarial attacks more difficult to implement, as the central server only sees the aggregate of the client updates. Therefore, it is of fundamental interest to investigate how to defend against adversarial attacks when secure aggregation is used. Existing approaches based on range proofs (e.g., Bulletproofs [328]) can guarantee that the DP-based clipping defense described above is compatible with SecAgg, but developing computation- and communication-efficient range proofs is still an active research direction.

SecAgg also introduces challenges for other defense methods. For example, many existing Byzantine-robust aggregation methods utilize non-linear operations on the server Xie *et al.* [421], and it is not yet known if these methods are efficiently compatible with secure aggregation which was originally designed for linear aggregation. Recent work has found ways to approximate the geometric median under SecAgg [420] by using a handful of SecAgg calls in a more general aggregation loop. However, it is not clear in general which aggregators can be computed under the use of SecAgg.

5.4 Executive Summary

- Third-party participants in the training process introduces new capabilities and attack vectors for adversaries, categorized in Table 5.1.
- Federated learning introduces a new kind of poisoning attacks, *model update poisoning* (Subsection 5.1.2), while also being susceptible to traditional *data poisoning* in (Subsection 5.1.3).
- Training participants can influence the optimization process possibly exacerbating *inference-time evasion attacks* (Subsection 5.1.4) and communication and computation constraints may render previously proposed defenses impractical.
- Non-malicious failure modes (Subsection 5.2) can be especially different to deal with, as access to raw data is not available in the federated setting, though through some lens they may be related to poisoning attacks.

- Tension may exist when trying to simultaneously improve robustness and privacy in machine learning (Subsection 5.3).

Areas identified for further exploration include:

- Quantify the relationship between data poisoning and model update poisoning attacks. Are there scenarios where they are not equivalent? [5.1.3]
- Quantify how training time attacks impact inference-time vulnerabilities. Improving inference-time robustness guarantees requires going beyond detecting backdoor attacks. [5.1.4]
- Adversarial training has been used as a defense in the centralized setting, but can be impractical in the edge-compute limited cross-device federated setting. [5.1.5]
- Federated learning requires new methods and tools to support the developer, as access to raw data is restricted debugging ML pipelines is especially difficult. [5.2]
- Tensions exists between robustness and fairness, as machine learning models can tend to discard updates far from the median as detrimental. However the federated setting can give rise to a long tail of users that may be mistaken for noisy model updates [5.2].
- Cryptography-based aggregation methods and robustness techniques present integration challenges: protecting participant identity can be at odds with detecting adversarial participants. Proposed techniques remain beyond the scope of practicality, requiring the need of new communication and computation efficient algorithms. [5.3]

6

Ensuring Fairness and Addressing Sources of Bias

Machine learning models can often exhibit surprising and unintended behaviors. When such behaviors lead to patterns of *undesirable* effects on users, we might categorize the model as “unfair” according to some criteria. For example, if people with similar characteristics receive quite different outcomes, then this violates the criterion of *individual fairness* [463]. If certain sensitive groups (races, genders, etc.) receive different patterns of outcomes—such as different false negative rates—this can violate various criteria of *demographic fairness*, see for instance [464], [465] for surveys. The criterion of *counterfactual fairness* requires that a user receive the same treatment as they would have if they had been a member of a different group (race, gender, etc.), after taking all causally relevant pathways into account [466].

Federated learning raises several opportunities for fairness research, some of which extend prior research directions in the non-federated setting, and others that are unique to federated learning. This section raises open problems in both categories.

6.1 Bias in Training Data

One driver of unfairness in machine-learned models is bias in the training data, including cognitive, sampling, reporting, and confirmation bias. One common antipattern is that minority or marginalized social groups are under-represented in the training data, and thus the learner weights these groups less during training [467], leading to inferior quality predictions for members of these groups (e.g., [468]).

Just as the data access processes used in federated learning may introduce dataset shift and non-independence (Subsection 3.1), there is also a risk of introducing biases. For example:

- If devices are selected for updates when plugged-in or fully charged, then model updates and evaluations computed at different times of day may be correlated with factors such as day-shift vs. night-shift work schedules.
- If devices are selected for updates from among the pool of eligible devices at a given time, then devices that are connected at times when few other devices are connected (e.g., night-shift or unusual time zone) may be over-represented in the aggregated output.
- If selected devices are more likely to have their output kept when the output is computed faster, then: (a) output from devices with faster processors may be over-represented, with these devices likely newer devices and thus correlated with socioeconomic status; and (b) devices with less data may be over-represented, with these devices possibly representing users who use the product less frequently.
- If data nodes have different amounts of data, then federated learning may weigh higher the contributions of populations which are heavy users of the product or feature generating the data.
- If the update frequency depends on latency, then certain geographic regions and populations with slower devices or networks may be under-represented.

- If populations of *potential users* do not own devices for socio-economic reasons, they may be under-represented in the training dataset, and subsequently also under- (or un-)represented in model training and evaluation.
- Unweighted aggregation of the model loss across selected devices during federated training may disadvantage model performance on certain devices [469].

It has been observed that biases in the data-generating process can also drive unfairness in the resulting models learned from this data (see e.g. [470], [471]). For example, suppose training data is based on user interactions with a product which has failed to incorporate inclusive design principles. Then, the user interactions with the product might not express user intents (cf. [472], for example) but rather might express coping strategies around uninclusive product designs (and hence might require a fundamental fix to the product interaction model). Learning from such interactions might then ignore or perpetuate poor experiences for some groups of product users in ways which can be difficult to detect while maintaining privacy in a federated setting. This risk is shared by all machine learning scenarios where training data is derived from user interaction, but is of particular note in the federated setting when data is collected from apps on individual devices.

Investigating the degree to which biases in the data-generated process can be identified or mitigated is a crucial problem for both federated learning research and ML research more broadly. Similarly, while limited prior research has demonstrated methods to identify and correct bias in already collected data in the federated setting (e.g., via adversarial methods in [473]), further research in this area is needed. Finally, methods for applying post-hoc fairness corrections to models learned from potentially biased training data are also a valuable direction for future work.

6.2 Fairness Without Access to Sensitive Attributes

Having explicit access to demographic information (race, gender, etc.) is critical to many existing fairness criteria, including those discussed in Subsection 6.1. However, the contexts in which federated learning are often deployed also give rise to considerations of fairness when individual sensitive attributes are *not* available. For example, this can occur when developing personalized language models or developing fair medical image classifiers without knowing any additional demographic information about individuals. Even more fundamentally, the assumed one-to-one relationship between individuals and devices often breaks down, especially in non-Western contexts [472]. Both measuring and correcting unfairness in contexts where there is no data regarding sensitive group membership is a key area for federated learning researchers to address.

Limited existing research has examined fairness without access to sensitive attributes. For example, this has been addressed using distributionally-robust optimization (DRO) which optimizes for the worst-case outcome across all individuals during training [474], and via multicalibration, which calibrates for fairness across subsets of the training data [475]. Even these existing approaches have not been applied in the federated setting, raising opportunities for future empirical work. The challenge of how to make these approaches work for large-scale, high-dimensional data typical to federated settings is also an open problem, as DRO and multicalibration both pose challenges of scaling with large n and p. Finally, the development of additional theoretical approaches to defining fairness without respect to "sensitive attributes" is a critical area for further research.

Other ways to approach this include reframing the existing notions of fairness, which are primarily concerned with equalizing the probability of an outcome (one of which is considered "positive" and another "negative" for the affected individual). Instead, fairness without access to sensitive attributes might be reframed as *equal access to effective models.* Under this interpretation of fairness, the goal is to maximize model utility across all individuals, regardless of their (unknown) demographic identities, and regardless of the "goodness“ of an individual outcome. Again, this matches the contexts in which federated learning is most commonly used, such as language modeling or medical image classification, where there is no clear notion of an outcome which is "good" for a user, and instead the aim is simply to make correct predictions for users, regardless of the outcome.

Existing federated learning research suggests possible ways to meet such an interpretation of fairness, e.g., via personalization [126], [139]. A similar conception of fairness, as “a more fair distribution of the model performance across devices”, is employed in [469].

The application of attribute-independent methods explicitly to ensure equitable model performance is an open opportunity for future federated learning research, and is particularly important as federated learning reaches maturity and sees increasing deployment with real populations of users without knowledge of their sensitive identities.

6.3 Fairness, Privacy, and Robustness

Fairness and data privacy seem to be complementary ethical concepts: in many of the real-world contexts where privacy protection is desired, fairness is also desired. Often this is due to the sensitivity of the underlying data. Because federated learning is most likely to be deployed in contexts of sensitive data where both privacy and fairness are desirable, it is important that FL research examines how FL might be able to address existing concerns about fairness in machine learning, and whether FL raises new fairness-related issues.

In some ways, however, the ideal of fairness seems to be in tension with the notions of privacy for which FL seeks to provide guarantees: differentially-private learning typically seeks to obscure individually-identifying characteristics, while fairness often requires knowing individuals’ membership in sensitive groups in order to measure or ensure fair predictions are being made. While the trade-off between differential privacy and fairness has been investigated in the non-federated setting [476], [477], there has been little work on how (or whether) FL may be able to uniquely address concerns about fairness.

Recent evidence suggesting that differentially-private learning can have disparate impact on sensitive subgroups [476]–[479] provides further motivation to investigate whether FL may be able to address such concerns. A potential solution to relax the tension between privacy (which aims to protect the model from being too dependent on individuals) and fairness (which encourages the model to perform well on under-represented classes) may be the application of techniques such as personalization (discussed in Subsection 3.3) and "hybrid differential privacy," where some users donate data with lesser privacy guarantees [311].

Furthermore, current differentially-private optimization schemes are applied without respect to sensitive attributes—from this perspective, it might be expected that empirical studies have shown evidence that differentially-private optimization impacts minority subgroups the most [478]. Modifications to differentially-private optimization algorithms which explicitly seek to preserve performance on minority subgroups, e.g., by adapting the noise and clipping mechanisms to account for the representation of groups within the data, would also likely do a great deal to limit potential disparate impacts of differentially-private modeling on minority subgroups in federated models trained with differential privacy. However, implementing such adaptive differentially-private mechanisms in a way that provides some form of privacy guarantee presents both algorithmic and theoretical challenges which need to be addressed by future work.

Further research is also needed to determine the extent to which the issues above arise in the federated setting. Furthermore, as noted in Subsection 6.2, the challenge of evaluating the impact of differential privacy on model fairness becomes particularly difficult when sensitive attributes are not available, as it is unclear how to identify subgroups for which a model is behaving badly and to quantify the "price" of differential privacy—investigating and addressing these challenges is an open problem for future work.

More broadly, one could more generally examine the relation between privacy, fairness, and *robustness* (see Section 5). Many previous works on machine learning, including federated learning, typically focus on isolated aspects of robustness (either against poisoning, or against evasion), privacy, or fairness. An important open challenge is to develop a joint understanding of federated learning systems that are robust, private, and fair. Such an integrated approach can provide opportunities to benefit from disparate but complementary mechanisms. Differential privacy mechanisms can be used to both mitigate data inference attacks, and provide a foundation for robustness against data poisoning. On the other hand, such an integrated approach also reveals new vulnerabilities. For example, recent work has revealed a trade-off between privacy and robustness against adversarial examples [480].

Finally, privacy and fairness naturally meet in the context of learning data representations that are independent of some sensitive attributes while preserving utility for a task of interest. Indeed, this objective can be motivated both in terms of privacy: to transform data so as to hide private attributes, and fairness: as a way to make models trained on such representations fair with respect to the attributes. In the centralized setting, one way to learn such representations is through adversarial training techniques, which have been applied to image and speech data [473], [481]–[484]. In the federated learning scenario, clients could apply the transformation locally to their data in order to enforce or improve privacy and/or fairness guarantees for the FL process. However, learning this transformation in a federated fashion (potentially under privacy and/or fairness constraints) is itself an open question.

6.4 Leveraging Federation to Improve Model Diversity

Federated learning presents the opportunity to integrate, through distributed training, datasets which may have previously been impractical or even illegal to combine in a single location. For example, the Health Insurance Portability and Accountability Act (HIPAA) and the Family Educational Rights and Privacy Act (FERPA) constrain the sharing of medical patient data and student educational data, respectively, in the United States. To date, these restrictions have led to modeling occurring in institutional silos: for example, using electronic health records or clinical images from individual medical institutions instead of pooling data and models across institutions [485], [486]. In contexts where membership in institutional datasets is correlated with individuals' specific sensitive attributes, or their behavior and outcomes more broadly, this can lead to poor representation for users in groups underrepresented at those institutions. Importantly, this lack of representation and diversity in the training data has been shown to lead to poor performance, e.g., in genetic disease models [487] and image classification models [468].

Federated learning presents an opportunity to leverage uniquely diverse datasets by providing efficient decentralized training protocols along with privacy and non-identifiability guarantees for the resulting models. This means that federated learning enables training on multi-institutional datasets in many domains where this was previously not possible. This provides a practical opportunity to leverage larger, more diverse datasets and explore the generalizability of models which were previously limited to small populations. More importantly, it provides an opportunity to improve the *fairness* of these models by combining data across boundaries which are likely to have been correlated with sensitive attributes. For instance, attendance at specific health or educational institutions may be correlated with individuals' ethnicity or socioeconomic status. As noted in Subsection 6.1 above, underrepresentation in training data is a proven driver of model unfairness.

Future federated learning research should investigate the degree to which improving diversity in a federated training setting also improves the fairness of the resulting model, and the degree to which the differential privacy mechanisms required in such settings may limit fairness and performance gains from increased diversity. This includes a need for both empirical research which applies federated learning and quantifies the interplay between diversity, fairness, privacy, and performance; along with theoretical research which provides a foundation for concepts such as diversity in the context of machine learning fairness.

6.5 Federated Fairness: New Opportunities and Challenges

It is important to note that federated learning provides unique opportunities and challenges for fairness researchers. For example, by allowing for datasets which are distributed both by observation, but even by features, federated learning can enable modeling and research using partitioned data which may be too sensitive to share directly [65], [78]. Increased availability of datasets which can be used in a federated manner can help to improve the diversity of training data available for machine learning models, which can advance fair modeling theory and practice.

Researchers and practitioners also need to address the unique fairness-related challenges created by federated learning. For example, federated learning can introduce new sources of bias through the decision of which clients to sample based on considerations such as connection type/quality, device type, location, activity patterns, and local dataset size [32]. Future work could investigate the degree to which these various sampling constraints affect the fairness of the resulting model, and how such impacts can be mitigated within the federated framework, e.g., [469], [488], [489]. Frameworks such as *agnostic federated learning* [149] provide approaches to control for bias in the training objective. Work to improve the fairness of existing federated training algorithms will be particularly important as advances begin to approach the technical limits of other components of FL systems, such as model compression, which initially helped to broaden the diversity of candidate clients during federated training processes. There is no unique fairness criterion generally adopted in the study of fairness, and multiple criteria have been proven to be mutually incompatible. One way to deal with this question is the *online fairness* framework and algorithms of Awasthi *et al.* [490]. Adapting such solutions to the federated learning setting and further improving upon them will be challenging research questions in ML fairness theory and algorithms.

In the classical centralized machine learning setting, a substantial amount of advancement has been made in the past decade to train fair classifiers, such as constrained optimization, post-shifting approaches, and distributionally-robust optimization [474], [491], [492]. It is an open question whether such approaches, which have demonstrated utility for improving fairness in centralized training, could be used under the setting of federated learning (and if so, under what additional assumptions) in which data are located in a decentralized fashion and practitioners may not obtain an unbiased sample of the data that match the distribution of the population.

6.6 Executive Summary

In addition to inheriting the already significant challenges related to bias, fairness, and privacy in centralized machine learning, federated learning also brings a new set of distinct challenges and opportunities in these areas. The importance of these considerations will likely continue to grow as the real-world deployment of FL expands to more users, domains, and applications.

- Bias in training data (Subsection 6.1) is a key consideration related to bias and fairness in FL models, particularly due to the additional sampling steps germane to federation (e.g., client sampling) and the transfer of some model computation to client devices.
- The lack of data regarding sensitive attributes in many FL deployments can pose challenges for measuring and ensuring fairness, and also suggests potential reframing of fairness problems in ways that do not require such data (Subsection 6.2).
- Since FL is often deployed in contexts which are both privacy- and fairness-sensitive, this can magnify tensions between privacy and fairness objectives in practice. Further work is needed to address the potential tension between methods which achieve privacy, fairness, and robustness in both federated and centralized learning (Subsection 6.3).
- Federated learning presents unique opportunities to improve the diversity of stakeholders and data incorporated into learning, which could improve both the overall quality of downstream models, as well as their fairness due to more representative datasets (Subsection 6.4).
- Federated learning presents fairness-related challenges not present in the centralized training regime, but also affords new solutions (Subsection 6.5).

7

Addressing System Challenges

As we will see in this section, the challenges in building systems for federated learning can be split fairly cleanly into the two separate settings of cross-device and cross-silo federated learning (see Subsections 1.1 and 2.2). We start with a brief discussion of the difficulties inherent to any large scale deployment of software on end-user devices (although exacerbated by the complexity of a federated learning stack); we then focus on key challenges specific to the cross-device learning—bias, tuning, and efficient device-side execution of ML workflows—before concluding with a brief treatment of the cross-silo setting.

7.1 Platform Development and Deployment Challenges

Running computations on end-user devices is considerably different from the data center setting:

- Due to the heterogeneity of the fleet (devices may differ in hardware, software, connectivity, performance and persisted state) the space of potential problems and edge cases is vast and cannot typically be covered in sufficient detail with automated testing.

- Monitoring and debugging are harder because telemetry is limited, delayed, and there is no physical access to devices for interactive troubleshooting.
- Running computations should not affect device performance or stability, i.e., should be invisible to users.

Code Deployment Installing, updating and running software on end user devices may involve not only extensive manual and automated testing, but a gradual and reversible rollout (for example, through guarding new functionality with server-controlled feature flags) while monitoring key performance metrics in a/b experiments such as crash rates, memory use, and application-dependent indicators such as latencies and engagement metrics. Such rollouts can take weeks or months depending on the percolation rate of updates (particularly challenging for devices with spotty connectivity) and the complexity of the upgrade (e.g., protocol changes). Hence, the install base at any given time will involve various releases. While this problem is not specific to federated learning, it has greater impact here due to the inherent collaborative nature of federated computations: devices constantly communicate with servers and indirectly with other devices to exchange models and parameter updates. Thus, compatibility concerns abound and must be addressed through stable exchange formats or, where not possible, detected upfront with extensive testing infrastructure. We will revisit this problem in Subsection 7.4.

Monitoring and Debugging Another significant complication is the limited ability to monitor devices and interactively debug problems. While telemetry from end user devices is necessary to detect problems, privacy concerns severely restrict what can be logged, who can access such logs, and how long they are retained. Once a regression is detected, drilling down into the root cause can be very cumbersome due to the lack of detailed context, the vast problem space (a cross product of software versions, hardware, models, and device state), and very limited ability for interactive debugging short of successfully reproducing the problem in a controlled environment.

These challenges are exacerbated in the federated learning setting where (a) raw input data on devices cannot be accessed, and (b) contributions from individual devices are by design anonymous, ephemeral, and exposed only in aggregate. These properties preserve privacy, but also may make it hard or impossible to investigate problems with traditional approaches—by looking for correlations with hardware or software version, or testing hypotheses that require access to raw data. Reproducing a problem in a controlled setting is often difficult due to the gap between such an environment and reality: hundreds of heterogeneous embedded stateful devices with non-IID data.

Interestingly, federated technologies themselves can help to mitigate this problem—for instance, the use of federated analytics [493] to collect logs in a privacy preserving manner, or training generative models of the system behavior or raw data for sampling during debugging (see Subsections 3.4.3, 5.2, and [176]). Keeping a federated learning system up and running thus requires investing into upfront detection of problems through (a) extensive automated, continuous test coverage of all software layers through both unit and integration tests; (b) feature flags and a/b rollouts; and (c) continuous monitoring of performance indicators for regressions. That poses a significant investment that may come at too high a cost for smaller entities who would benefit greatly from shared and tested infrastructure for federated learning.

7.2 System Induced Bias

Deployment, monitoring and debugging may not concern users of a federated learning platform, e.g., model authors or data analysts. For them, the key differences between data center and cross-device settings fall largely into the following two categories:

1. **Availability of devices** for computations is not a given, but varies over time and across devices. Connections are initiated by devices and subject to interruptions due to changes in device state, operating system quotas, and network connectivity. Hence, in iterative processes like federated learning, the loop body is run

on a small subset of all devices only, and the system must tolerate a certain failure rate among those devices.

2. **Capabilities of devices** (network bandwidth and latency, compute performance, memory) vary, and are typically much lower than those of compute nodes in the data center, though the number of nodes is typically higher. The amount and type of data across devices may lead to variations in execution profile, e.g., more and larger examples lead to increased resource use and processing time.

In the following sections we discuss how these variations might introduce bias, referring to it as system induced bias to differentiate it from platform-independent bias in the raw data (such as ownership or usage patterns differing across demographics)—for the latter, see Subsection 6.1.

7.2.1 Device Availability Profiles

At the core of cross-device federated learning is the principle that devices only connect to the server and run computations when various constraints are met:

- **Hard constraints**, which might include requiring that the device is turned on, has network connectivity to the server, and is allowed to run a computation by the operating system.
- **Soft constraints**, which might include the conditions on device state chosen to ensure that federated learning does not incur charges or affect usability. For the common case of mobile phones [17], [32], requirements may include idleness, charging and/or above a certain battery level, being connected to an unmetered network, and that no other federated learning tasks are running at the same time.

Taken together, these constraints induce an unknown, time-varying and device-specific function $A_i(t)$ for a device i, and a fleet-wide *availability profile* $A(t) = \sum_i A_i(t)$. Round completion rates and server traffic patterns [14], [32] suggest that availability profiles for mobile phones are clustered into periodic functions with a period of 1 day, varying across devices in phase, shape and amplitude through factors such as demographics, geography etc. Availability for other end user devices such as laptops, tablets, or stationary devices such as smart speakers, displays and cameras, will differ, but the challenges discussed in the following sections apply there as well, albeit to a possibly lesser extent.

7.2.2 Examples of System Induced Bias

Sources of bias will depend on the specific way in which devices are selected to participate in training, and how the system influences which devices end up contributing to the final aggregated model update. Thus, it is useful to discuss these issues in light of a simplified but representative system design. In an iterative federated learning algorithm, such as Federated Averaging (Subsection 1.1.2, [1]), rounds are run consecutively on sets of at least M devices. To accommodate a fraction d of devices not contributing due to changes in device conditions, time-outs, or slowness (server-side aborts to avoid slow-downs by stragglers), an over-allocation scheme is used where

1. Rounds are started when at least $M' = \frac{M}{1-d}$ devices are available.
2. Rounds are closed as
   (a) *Aborted* when more than $M' - M$ devices have disconnected, or
   (b) *Successful* when at least M devices have reported. One possible design choice is to stop after exactly M devices; another possibility would be to keep waiting for stragglers (possibly up to some maximum time).

This sequence, when combined with variable availability profiles, may introduce various forms of bias:

1. Selection Bias—whether a device is included in a round at time t depends on both
   (a) Its availability profile $A_i(t)$.

   (b) The number of simultaneously connected devices: $< M'$ and a round cannot be started; $\gg M'$ and the probability of a single device being included becomes very small. In effect, devices active only at either fleet-wide availability peaks or troughs may be under-represented.

2. Survival Bias
   (a) Since a server might choose to close a round at any point after the first M devices have reported, contributions are biased towards devices with better network connections, faster processors, lower CPU load, and less data to process.
   (b) Devices drop out of rounds when they are interrupted by the operating system, which may happen due to changes in device conditions as described by $A_i(t)$, or due to e.g., excessive memory use.

As can be seen, the probability of a device contributing to a round of federated learning is a complex function of both internal (e.g., device specific) and external (fleet dynamic) factors. When this probability is correlated with statistics of the data distribution, aggregate results may be biased. For instance, language models may over-represent demographics that have high quality internet connections or high end devices; and ranking models may not incorporate enough contributions from high engagement users who produce a lot of training data and hence longer training times.

Thus, designing systems that explicitly take such factors into account and integrate algorithms designed to both quantify and mitigate these effects are a fundamentally important research direction.

7.2.3 Open Challenges in Quantifying and Mitigating System Induced Bias

While the potential for bias in federated learning has been addressed in the literature (Section 6, [32], [92], [469]), a systematic study that qualifies and quantifies bias in realistic settings and its sources is a direction for future research. Conducting the necessary work may be hampered by both access to the necessary resources, and the difficulty in quantifying bias in a final statistical estimate due to the inherent lack of ground truth value.

We want to encourage further research to study how bias can be quantified and subsequently mitigated. A useful proxy metric for bias is to study the expected rate of contribution of a device to federated learning. In an unbiased system, this rate would be identical for every device; if it is not, the non-uniformity may provide a measure of bias. Studying the root causes for this non-uniformity may then provide important hints for how to mitigate bias, for example:

- When there is a strong correlation between devices finishing a round, and the number of examples they process or model size, possible fixes may include early stopping, or decreasing the model size.

- If the expected rate of contribution depends on factors outside our control, such as device model, network connectivity, location etc., one can view these factors as defining strata and applying *post-stratification* [494], that is, correcting for bias by scaling up or down contributions from devices depending on their stratum. It may also be possible to apply *stratified sampling*—e.g., change scheduling, or server selection policies, to affect the probability of including devices in a round as a function of their stratum.

- A very general, root-cause-agnostic mitigation could base the weight of a contribution solely on a device's past contribution profile (e.g., the number of rounds started or completed thus far). As a special case, consider *sampling without replacement* which could be implemented at the system level (stop connecting after one successful contribution) or at the model level (weight all but the first contribution with 0). This approach might not be sufficient when a population is large enough for most devices to contribute only infrequently (mostly one or zero times); in such cases, clustering devices based on some similarity metric and using cluster membership as stratum could help.

- Alternatives to the synchronous, round based execution described in the previous section may also help to mitigate bias. In particular, certain types of analytics may benefit from softening or eliminating the competition between devices for inclusion, by running rounds for long times with very large numbers of participants and without applying time-outs to stragglers. Such a method may not be applicable to algorithms where the iterative aspect (running many individual, chained rounds) is important.

The biggest obstacle to enabling such research is access to a representative fleet of end user devices, or a detailed description (e.g., in the form of a statistical model of a realistic distribution over $A_i(t)$ functions) of a fleet that can be used in simulations. Here, maintainers of FL production stacks are uniquely positioned to provide such statistics or models to academic partners in a privacy preserving fashion; a further promising direction is the recent introduction of the Flower framework [495] for federated learning research.

7.3 System Parameter Tuning

Practical federated learning is a form of multi-objective optimization: while the first order goal is maximizing model quality metrics such as loss or accuracy, other important considerations are:

- Convergence speed.
- Throughput (e.g., number of rounds, amount of data, or number of devices).
- Model fairness, privacy, and robustness (see Subsection 6.3).
- Resource use on server and clients.

These goals may be in tension. For instance, maximizing round throughput may introduce bias or hurt accuracy by preferring performant devices with little or no data. Maximizing for low training loss by increasing model complexity will put devices with less memory, many or large examples, or slow CPUs at a disadvantage. Bias or fairness induced in such a way during training may be hard to detect in the evaluation phase since it typically uses the same platform and hence is subject to similar biases.

Various controls affect the above listed indicators. Some are familiar from the datacenter setting, in particular model specific settings and learning algorithm hyperparameters. Others are specific to federated learning:

- **Clients per round**: The minimum number of devices required to complete a round, M, and the number of devices required to start a round, M'.

- **Server-side scheduling**: In all but the simplest cases, a federated learning system will operate on more than one model at a time: to support multiple tenants; to train models on the same data for different use cases; to support experimentation and architecture or hyper-parameter grid search; and to run training and evaluation workloads concurrently. The server needs to decide which task to serve to incoming devices, an instance of a scheduling problem: assigning work (training or evaluation tasks) to resources (devices). Accordingly, the usual challenges arise: ideal resource assignment should be fair, avoid starvation, minimize wait times, and support relative priorities all at once.

- **Device-side scheduling**: As described in Subsection 7.2, various constraints govern when a device can connect to the server and execute work. Within these constraints, various scheduling choices can be made. One extreme is to connect to the server and run computations as often as possible, leading to high load and resource use on both server and devices. Another choice are fixed intervals, but they need to be adjusted to reflect external factors such as number of devices overall and per round. The federated learning system developed at Google aims to strike a balance with a flow control mechanism called *pace steering* [32] whereby the server instructs devices when to return. Such a dynamic system enables temporal load balancing for large populations as well as "focusing" connection attempts to specific points in time to reach the threshold M'. Developing such a mechanism is difficult due to stochastic and dynamic nature of device availability, the lack of a predictive model of population behavior, and feedback loops.

Defining reasonable composite objective functions, and designing algorithms to automatically tune these settings, has not been explored yet in the context of federated learning systems and hence remains a topic of future research.

7.4 On-Device Runtime

While numerous frameworks exist for data center training, the options for training models on resource constrained devices are fairly limited. Machine Learning models and training procedures are typically authored in a high level language such as Python. For federated learning, this description encompasses device and server computations that are executed on the target platform and exchange data over a network connection, necessitating

- A means of serializing and dynamically transmitting local pieces of the total computation (e.g., the server-side update to the model, or the local client training procedure).
- A means to interpret or execute such a computation on the target platform (server or device).
- A stable network protocol for data exchange between participating devices and servers.

One extreme form of a representation is the original high-level description, e.g., a Python TensorFlow program [34]. This would require a Python interpreter with TensorFlow backend, which may not be a feasible choice for end-user devices due to resource constraints (binary size, memory use), performance limitations, or security concerns.

Another extreme representation of a computation is machine code of the target architecture, e.g., ARM64 instructions. This requires a compiler or re-implementation of a model in a lower-level language such as C++, and deployment computations will typically be subject to the restrictions that apply to deployment of binary code (see Subsection 7.1), introducing prohibitive latencies for executing novel computations.

Intermediate representations that can be compiled or interpreted with a runtime on the target platform strike a balance between flexibility and efficiency. However, such runtimes are currently not widely available. For instance, Google's FL system [32] relies on TensorFlow for both server and device side execution as well as model and parameter transfer, but this choice suffers from several shortcomings:

- It offers no easy path to devices for alternative front ends such as PyTorch [496], JAX [497] or CNTK [498].
- The runtime is not developed or optimized for resource constrained environments, incurring a large binary size, high memory use and comparatively low performance.
- The intermediate representation `GraphDef` used by TensorFlow is not standardized or stable, and version skew between the frontend and older on-device backends causes frequent compatibility challenges.

Other alternatives include more specialized runtimes that support only a subset of the frontend's capabilities, for instance training specific model types only, requiring changes and long update cycles whenever new model architectures or training algorithms are to be used. An extreme case would be a runtime that is limited and optimized to train a single type of model.

An ideal on-device runtime would have the following characteristics:

1. Lightweight: small binary size, or pre-installed; low memory and power profile.
2. Performant: low startup latency; high throughput, supports hardware acceleration.
3. Expressive: supports common data types and computations including backpropagation, variables, control flow, custom extensions.
4. Stable and compact format for expressing data and computations.

5. Widely available: portable open source implementation.
6. Targetable by commonly used ML frameworks/languages.
7. Ideally also supports inference, or if not, building personalized models for an inference runtime.

To our best knowledge no solution exists yet that satisfies these requirements, and we expect the limited ability to run ML training on end user devices to become a hindrance to adoption of federated technologies.

7.5 The Cross-Silo Setting

The system challenges arising in the scenario of cross-silo federated learning take a considerably different form. As outlined in Table 1.1, clients are fewer in number, more powerful, reliable, and known/addressable, eliminating many of the challenges from the cross-device setting, while allowing for authentication and verification, accounting, and contractually enforced penalties for misbehavior. Nonetheless, there are other sources of heterogeneity, including the features and distribution of data, and possibly the software stack used for training.

While the infrastructure in the cross-device setting (from the device-side data generation to the server logic) is typically operated by one or few organizational entities (the application, operating system, or device manufacturer), in the cross-silo setting, many different entities are involved. This may lead to high coordination and operational cost due to differences in:

- *How data is generated, pre-processed and labeled.* Learning across silos will require data normalization which may be difficult when such data is collected and stored differently (e.g., use of different medical imaging systems, and inconsistencies in labeling procedures, annotations, and storage formats).
- *Which software at which version powers training.* Using the same software stack in every silo—possibly delivered alongside the model using container technologies as done by FATE [27]—eliminates

compatibility concerns, but such frequent and centrally distributed software delivery may not be acceptable to all involved parties. An alternative that is more similar to the cross-device setting would be to standardize data and model formats and communication protocols. See IEEE P3652.1 "Federated Machine Learning Working Group" for a related effort in this direction.

- *The approval process for how data may or may not be used.* While this process is typically centralized in the cross-device scenario, the situation is likely different in cross-silo settings where many organizational entities are involved, and may be increasingly difficult when training spans different jurisdictions with varying data protection regulations. Technical infrastructure may be of help here by establishing data annotations that encode access policies, and infrastructure enforce them; for instance, limiting the use of certain data to specific models, or encoding minimum aggregation requirements such as "require at least M clients per round".

Another potential difference in the cross-silo setting is data partitioning: Data in the cross-device setting is typically assumed to be partitioned by examples, all of which have the same features (horizontal partitioning). In the cross-silo setting, in addition to partitioning by examples, partitioning by features is of practical relevance (vertical partitioning). An example would be two organizations, e.g., a bank and a retail company, with an overlapping set of customers, but different information (features) associated with them. For a discussion focusing on the algorithmic aspects, please see Subsection 2.2. Learning with feature-partitioned data may require different communication patterns and additional processing steps e.g., for entity alignment and dealing with missing features.

7.6 Executive Summary

While production grade systems for cross-device federated learning operate successfully [17], [32], various challenges remain:

- Frequent and large scale deployment of updates, monitoring, and debugging is challenging (Subsection 7.1).

- Differences in device availability induce various forms of bias; defining, quantifying and mitigating them remains a direction for future research (Subsection 7.2).
- Tuning system parameters is difficult due to the existence of multiple, potentially conflicting objectives (Subsection 7.3).
- Running ML workloads on end user devices is hampered by the lack of a portable, fast, small footprint, and flexible runtime for on-device training (Subsection 7.4).

Systems for cross-silo settings (Subsection 7.5) face largely different issues owing to differences in the capabilities of compute nodes and the nature of the data being processed.

8

Concluding Remarks

Federated learning enables distributed client devices to collaboratively learn a shared prediction model while keeping all the training data on device, decoupling the ability to do machine learning from the need to store the data in the cloud. This goes beyond the use of local models that make predictions on mobile devices by bringing model training to the device as well.

In recent years, this topic has undergone an explosive growth of interest, both in industry and academia. Major technology companies have already deployed federated learning in production, and a number of startups were founded with the objective of using federated learning to address privacy and data collection challenges in various industries. Further, the breadth of papers surveyed in this work suggests that federated learning is gaining traction in a wide range of interdisciplinary fields: from machine learning to optimization to information theory and statistics to cryptography, fairness, and privacy.

Motivated by the growing interest in federated learning research, this monograph discusses recent advances and presents an extensive collection of open problems and challenges. The system constraints impose efficiency requirements on the algorithms in order to be practical, many of which are not particularly challenging in other settings. We argue that data privacy is not binary and present a range of threat models that are relevant under a variety of assumptions, each of which provides its own unique challenges.

The open problems discussed in this work are certainly not comprehensive, they reflect the interests and backgrounds of the authors. In particular, we do not discuss any non-learning problems which need to be solved in the course of a practical machine learning project, and might need to be solved based on decentralized data [493]. This can include simple problems such as computing basic descriptive statistics, or more complex objectives such as computing the head of a histogram over an open set [499]. Existing algorithms for solving such problems often do not always have an obvious "federated version" that would be efficient under the system assumptions motivating this work or do not admit a useful notion of data protection. Yet another set of important topics that were not discussed are the legal and business issues that may motivate or constrain the use of federated learning.

We hope this work will be helpful in scoping further research in federated learning and related areas.

Acknowledgments

The authors would like to thank Alex Ingerman and David Petrou for their useful suggestions and insightful comments during the review process.

Appendices

A.1 Software and Datasets for Federated Learning

Software for Simulation Simulations of federated learning require dealing with multiple issues that do not arise in datacenter ML research, for example, efficiently processing partitioned datasets, with computations running on different simulated devices, each with a variable amount of data. FL research also requires different metrics such as the number of bytes upload or downloaded by device, as well as the ability to simulate issues like time-varying arrival of different clients or client drop-out that is potentially correlated with the nature of the local dataset. With this in mind, the development of open software frameworks for federated learning research (simulation) has the potential to greatly accelerate research progress. Several platforms are available or in development, including [248]:

- TensorFlow Federated [26] specifically targets research use cases, providing large-scale simulation capabilities as well as flexible orchestration for the control of sampling.
- FedML [500] is a research-oriented library. It supports three platforms: on-device training for IoT and mobile devices, distributed computing, and single-machine simulation. For research diversity, FedML also supports various algorithms (e.g., decentralized learning, vertical FL, and split learning), models, and datasets.
- PySyft [28] is a Python library for secure, private Deep Learning. PySyft decouples private data from model training, using federated learning, differential privacy, and multi-party computation (MPC) within PyTorch.
- Leaf [29] provides multiple datasets (see below), as well as simulation and evaluation capabilities.
- Sherpa.ai Federated Learning and Differential Privacy Framework [501] is an open source federated learning and differential privacy framework which provides methodologies, pipelines, and evaluation techniques for federated learning.

- PyVertical [502] is a project focusing on federated learning with data partitioned by features (also referred to as vertical partitioning) in the cross-silo setting; see Subsection 2.2.

Production-Oriented Software In addition to the above simulation platforms, several production-oriented federated learning platforms are being developed:

- FATE (Federated AI Technology Enabler) [27] is an open-source project intended to provide a secure computing framework to support the federated AI ecosystem.
- PaddleFL [30] is an open source federated learning framework based on PaddlePaddle [503]. In PaddleFL, several federated learning strategies and training strategies are provided with application demonstrations.
- Clara Training Framework [31] includes the support of cross-silo federated learning based on a server-client approach with data privacy protection.
- IBM Federated Learning [71] is a Python-based federated learning framework for enterprise environments, which provides a basic fabric for adding advanced features.
- Flower framework [495] supports implementation and experimentation of federated learning algorithms on mobile and embedded devices with a real-world system conditions simulation.
- Fedlearner [504] is an open source federated learning framework that enables joint modeling of data distributed between institutions.

Such production-oriented federated learning platforms must address problems that do not exist in simulation such as authentication, communication protocols, encryption and deployment to physical devices or silos. Note that while TensorFlow Federated is listed under "Software for simulation", its design includes abstractions for aggregation and broadcast, and serialization of all TensorFlow computations for execution in non-Python environments, making it suitable for use as a component in a production system.

Datasets Federated learning is adopted when the data is decentralized and typically unbalanced (different clients have different numbers of examples) and not identically distributed (each client's data is drawn from a different distribution). The open source package TensorFlow Federated [26] supports loading decentralized dataset in a simulated environment with each client id corresponding to a TensorFlow Dataset Object. These datasets can easily be converted to numpy arrays for use in other frameworks.[1] At the time of writing, three datasets are supported and we recommend researchers to benchmark on them.

- *EMNIST* dataset [505] consists of 671,585 images of digits and upper and lower case English characters (62 classes). The federated version splits the dataset into 3,400 unbalanced clients indexed by the original writer of the digits/characters. The non-IID distribution comes from the unique writing style of each person.
- *Stackoverflow*[2] dataset consists of question and answer from Stack Overflow with metadata like timestamps, scores, etc. The training dataset has more than 342,477 unique users with 135,818,730 examples. Note that the timestamp information can be helpful to simulate the pattern of incoming data.
- *Shakespeare* is a language modeling dataset derived from *The Complete Works of William Shakespeare*. It consists of 715 characters whose contiguous lines are examples in the client dataset. The train set has 16,068 examples and test set has 2,356 examples.

The preprocessing for *EMNIST* and *Shakespeare* are provided by the Leaf project [506], which also provides federated versions of the sentiment140 and celebA datasets. These datasets have enough clients that they can be used to simulate cross-device FL scenarios, but for

[1] https://www.tensorflow.org/datasets/api_docs/python/tfds/as_numpy.
[2] https://www.kaggle.com/stackoverflow/stackoverflow.

questions where scale is particularly important, they may be too small. In this respect *Stackoverflow* provides the most realistic example of a cross-device FL problem.

Cross-Silo Datasets One example is the iNaturalist dataset[3] which consists of large numbers of observations of various organisms all over the world. One can partition it by the geolocation or the author of an observation. If we partition it by the group an organism belongs to, like kingdom, phylum, etc., then the clients have totally different labels and biological closeness between two clients is already known. This makes it a very suitable dataset to study federated transfer learning and multi-task learning in cross-silo settings.

Another example is the Google-Landmark-v2 [507] that includes over five million images of more than 200 thousand different types of landmark. Similar to the iNaturalist dataset, one can split the dataset by authors, but due to the difference in scale with iNaturalist dataset, Google Landmark Dataset provides much more diversity and creates even greater challenges to large-scale federated learning.

Luo *et al.* [508] has recently published a federated dataset for computer vision. The dataset contains more than 900 annotated street images generated from 26 street cameras and seven object categories annotated with detailed bounding box. Due to the relatively small number of examples in the dataset, it may not adequately reflect a challenging realistic scenario.

The Need for More Datasets Developing new federated learning datasets that are representative of real-world problems is an important question for the community to address. Platforms like TensorFlow Federated [26] welcome the contribution of new datasets and may be able to provide hosting support.

While completely new datasets are always interesting, in many cases it is possible to partition existing open datasets, treating each split as a client. Different partitioning strategies may be appropriate for different research questions, but often unbalanced and non-IID partitions will

[3] https://www.inaturalist.org/.

be most relevant. It is also interesting to maintain as much additional meta information (timestamp, geolocation, etc.) as possible.

In particular, there is a need for feature-partitioned datasets, as will be discussed in Subsection 2.2. For example, a patient may go to one medical institute for a pathology test and go to another for radiology picture archiving, in which case the features of one sample are partitioned over two institutes regulated by HIPAA [509].

References

[1] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, pp. 1273–1282, 2017.

[2] R. L. Rivest, L. Adleman, and M. L. Dertouzos, "On data banks and privacy homomorphisms," *Foundations of Secure Computation, Academia Press*, pp. 169–179, 1978.

[3] A. C. Yao, "Protocols for secure computations," in *Symposium on Foundations of Computer Science*, pp. 160–164, 1982.

[4] R. Agrawal and R. Srikant, "Privacy-preserving data mining," in *ACM SIGMOD International Conference on Management of Data*, pp. 439–450, 2000.

[5] J. Vaidya, H. Yu, and X. Jiang, "Privacy-preserving SVM classification," *Knowl. Inf. Syst.*, vol. 14, no. 2, pp. 161–178, 2008.

[6] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Process. Mag.*, vol. 37, no. 3, pp. 50–60, 2020.

[7] H. B. McMahan and D. Ramage, *Federated learning: Collaborative machine learning without centralized training data*, [Online]. Available: https://ai.googleblog.com/2017/04/federated-learning-collaborative.html. Google AI Blog, Apr. 2017.

[8] Differential Privacy Team, "Learning with privacy at scale," *Apple Machine Learning Journal*, vol. 1, no. 8, 2017. [Online]. Available: https://machinelearning.apple.com/2017/12/06/learning-with-privacy-at-scale.html.

[9] Ú. Erlingsson, V. Pihur, and A. Korolova, "RAPPOR: Randomized aggregatable privacy-preserving ordinal response," in *ACM CCS,* 2014. DOI: 10.1145/2660267.2660348.

[10] M. Chen, R. Mathews, T. Ouyang, and F. Beaufays, "Federated learning of out-of-vocabulary words," *arXiv preprint 1903.10635*, 2019. [Online]. Available: http://arxiv.org/abs/1903.10635.

[11] A. Hard, K. Rao, R. Mathews, F. Beaufays, S. Augenstein, H. Eichner, C. Kiddon, and D. Ramage, "Federated learning for mobile keyboard prediction," *arXiv preprint 1811.03604*, 2018.

[12] S. Pichai, "Google's Sundar Pichai: Privacy should not be a luxury good," *New York Times*, May 7, 2019.

[13] S. Ramaswamy, R. Mathews, K. Rao, and F. Beaufays, "Federated learning for emoji prediction in a mobile keyboard," *arXiv preprint arxiv:1906.04329*, 2019.

[14] T. Yang, G. Andrew, H. Eichner, H. Sun, W. Li, N. Kong, D. Ramage, and F. Beaufays, "Applied federated learning: Improving Google keyboard query suggestions," *arXiv preprint arxiv:1812.02903*, 2018.

[15] support.google, *Your chats stay private while Messages improves suggestions*, 2019. [Online]. Available: https://support.google.com/messages/answer/9327902. Retrieved Aug. 2019.

[16] Apple, *Private federated learning (NeurIPS 2019 Expo Talk Abstract)*, https://nips.cc/ExpoConferences/2019/schedule?talk_id=40, 2019.

[17] Apple, *Designing for privacy* (video and slide deck), Apple WWDC, https://developer.apple.com/videos/play/wwdc2019/708, 2019.

[18] W. de Brouwer, *The federated future is ready for shipping*, https://medium.com/@_doc_ai/the-federated-future-is-ready-for-shipping-d17ff40f43e3, Mar. 2019.

[19] D. Leroy, A. Coucke, T. Lavril, T. Gisselbrecht, and J. Dureau, “Federated learning for keyword spotting,” *arXiv preprint arXiv: 1810.05512*, 2018.

[20] WeBank, *WeBank and Swiss re signed cooperation MoU*, 2019. [Online]. Available: https://www.fedai.org/news/webank-and-swiss-re-signed-cooperation-mou/. Retrieved Aug. 2019.

[21] EU CORDIS, *Machine learning ledger orchestration for drug discovery*, 2019. [Online]. Available: https://cordis.europa.eu/project/rcn/223634/factsheet/en?WT.mc_id=RSS-Feed&WT.rss_f=project&WT.rss_a=223634&WT.rss_ev=a. Retrieved Aug. 2019.

[22] FeatureCloud, *FeatureCloud: Our vision*, 2019. [Online]. Available: https://featurecloud.eu/about/our-vision/. Retrieved Aug. 2019.

[23] ai.intel, *Federated learning for medical imaging*, 2019. [Online]. Available: https://www.intel.ai/federated-learning-for-medical-imaging/. Retrieved Aug. 2019.

[24] P. Courtiol, C. Maussion, M. Moarii, E. Pronier, S. Pilcer, M. Sefta, P. Manceron, S. Toldo, M. Zaslavskiy, N. Le Stang, N. Girard, O. Elemento, A. G. Nicholson, J.-Y. Blay, F. Galateau-Sallé, G. Wainrib, and T. Clozel, “Deep learning-based classification of mesothelioma improves prediction of patient outcome,” *Nature Medicine*, pp. 1–7, 2019.

[25] Musketeer, *Musketeer: About*, 2019. [Online]. Available: http://musketeer.eu/project/. Retrieved Aug. 2019.

[26] The TFF Authors, *TensorFlow Federated*, 2019. [Online]. Available: https://www.tensorflow.org/federated.

[27] The FATE Authors, *Federated AI technology enabler*, 2019. [Online]. Available: https://www.fedai.org/.

[28] T. Ryffel, A. Trask, M. Dahl, B. Wagner, J. Mancuso, D. Rueckert, and J. Passerat-Palmbach, “A generic framework for privacy preserving deep learning,” *arXiv preprint arXiv:1811.04017*, 2018.

[29] The Leaf Authors, *Leaf*, 2019. [Online]. Available: https://leaf.cmu.edu/.

[30] The PaddleFL Authors, *PaddleFL*, 2019. [Online]. Available: https://github.com/PaddlePaddle/PaddleFL.

[31] N. Clara, *The clara training framework authors*, 2019. [Online]. Available: https://developer.nvidia.com/clara.

[32] K. A. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. M. Kiddon, J. Konečný, S. Mazzocchi, B. McMahan, T. V. Overveldt, D. Petrou, D. Ramage, and J. Roselander, "Towards federated learning at scale: System design," in *SysML 2019*, 2019. [Online]. Available: https://arxiv.org/abs/1902.01046.

[33] J. Dean, G. S. Corrado, R. Monga, K. Chen, M. Devin, Q. V. Le, M. Z. Mao, M. Ranzato, A. Senior, P. Tucker, K. Yang, and A. Y. Ng, "Large scale distributed deep networks," in *Proceedings of the International Conference on Neural Information Processing Systems*, pp. 1223–1231, 2012.

[34] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, *TensorFlow: Large-scale machine learning on heterogeneous systems*, Software available from tensorflow.org, 2015. [Online]. Available: https://www.tensorflow.org/.

[35] P. Vanhaesebrouck, A. Bellet, and M. Tommasi, "Decentralized collaborative learning of personalized models over networks," in *AISTATS*, 2017.

[36] X. Lian, C. Zhang, H. Zhang, C.-J. Hsieh, W. Zhang, and J. Liu, "Can decentralized algorithms outperform centralized algorithms? A case study for decentralized parallel stochastic gradient descent," in *NIPS*, 2017.

[37] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Randomized gossip algorithms," *IEEE Trans. Inform. Theor.*, vol. 52, no. 6, pp. 2508–2530, 2006.

[38] M. Assran, N. Loizou, N. Ballas, and M. Rabbat, "Stochastic gradient push for distributed deep learning," in *ICML*, 2019.

[39] A. Bellet, R. Guerraoui, M. Taziki, and M. Tommasi, "Personalized and Private Peer-to-Peer Machine Learning," in *AISTATS*, 2018.

[40] I. Colin, A. Bellet, J. Salmon, and S. Clémençon, "Gossip dual averaging for decentralized optimization of pairwise functions," in *ICML*, 2016.

[41] A. Elgabli, J. Park, A. S. Bedi, M. Bennis, and V. Aggarwal, "GADMM: Fast and communication efficient framework for distributed machine learning," *arXiv preprint arXiv:1909.00047*, 2019.

[42] A. Koloskova, S. U. Stich, and M. Jaggi, "Decentralized stochastic optimization and Gossip algorithms with compressed communication," in *ICML*, 2019.

[43] A. Lalitha, O. C. Kilinc, T. Javidi, and F. Koushanfar, "Peer-to-peer federated learning on graphs," arXiv:1901.11173, Tech. Rep., 2019.

[44] H. Tang, X. Lian, M. Yan, C. Zhang, and J. Liu, "D2: Decentralized training over decentralized data," in *ICML*, 2018.

[45] C. He, C. Tan, H. Tang, S. Qiu, and J. Liu, "Central server free federated learning over single-sided trust social networks," *arXiv preprint arXiv:1910.04956*, 2019.

[46] L. He, A. Bian, and M. Jaggi, "COLA: Decentralized linear learning," in *NeurIPS 2018 – Advances in Neural Information Processing Systems*, vol. 31, pp. 4541–4551, 2018.

[47] C. Yu, H. Tang, C. Renggli, S. Kassing, A. Singla, D. Alistarh, C. Zhang, and J. Liu, "Distributed learning over unreliable networks," *arXiv preprint arXiv:1810.07766*, 2018.

[48] G. Neglia, C. Xu, D. Towsley, and G. Calbi, "Decentralized gradient methods: Does topology matter?" In *AISTATS*, 2020.

[49] J. Wang, A. Sahu, G. Joshi, and S. Kar, "MATCHA: Speeding up decentralized SGD via matching decomposition sampling," *Preprint*, May 2019. [Online]. Available: https://arxiv.org/abs/1905.09435.

[50] X. Lian, W. Zhang, C. Zhang, and J. Liu, "Asynchronous decentralized parallel stochastic gradient descent," in *ICML*, 2018.

[51] A. Koloskova, T. Lin, S. U. Stich, and M. Jaggi, "Decentralized deep learning with arbitrary communication compression," *International Conference on Learning Representations (ICLR)*, 2020.

[52] A. Koloskova, N. Loizou, S. Boreiri, M. Jaggi, and S. U. Stich, "A unified theory of decentralized SGD with changing topology and local updates," in *ICML*, 2020.

[53] J. Wang and G. Joshi, "Cooperative SGD: A unified framework for the design and analysis of communication-efficient SGD algorithms," *Preprint,* Aug., 2018. [Online]. Available: https://arxiv.org/abs/1808.07576.

[54] V. Zantedeschi, A. Bellet, and M. Tommasi, *Fully decentralized joint learning of personalized models and collaboration graphs*, Tech. Rep., arXiv:1901.08460, 2019.

[55] A. Reisizadeh, H. Taheri, A. Mokhtari, H. Hassani, and R. Pedarsani, "Robust and communication-efficient collaborative learning," *arXiv:1907.10595*, 2019.

[56] H. Tang, X. Lian, S. Qiu, L. Yuan, C. Zhang, T. Zhang, and J. Liu, "DeepSqueeze: Parallel stochastic gradient descent with double-pass error-compensated compression," *arXiv preprint arXiv:1907.07346*, 2019.

[57] Z. Huang, S. Mitra, and N. Vaidya, "Differentially private distributed optimization," in *ICDCN*, 2015.

[58] E. Cyffers and A. Bellet, "Privacy amplification by decentralization," *arXiv preprint arXiv:2012.05326*, 2020.

[59] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum Project Yellow Paper*, vol. 151, no. 2014, pp. 1–32, 2014.

[60] 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 *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, ACM, pp. 1175–1191, 2017.

[61] R. Cheng, F. Zhang, J. Kos, W. He, N. Hynes, N. Johnson, A. Juels, A. Miller, and D. Song, "Ekiden: A platform for confidentiality-preserving, trustworthy, and performant smart contracts," in *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*, pp. 185–200, IEEE, 2019.

[62] H. B. McMahan, D. Ramage, K. Talwar, and L. Zhang, "Learning differentially private recurrent language models," in *International Conference on Learning Representations (ICLR)*, 2018.

[63] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *CoRR*, vol. abs/1902.04885, 2019. [Online]. Available: http://arxiv.org/abs/1902.04885.

[64] K. Cheng, T. Fan, Y. Jin, Y. Liu, T. Chen, and Q. Yang, "SecureBoost: A lossless federated learning framework," *CoRR*, vol. abs/1901.08755, 2019. [Online]. Available: http://arxiv.org/abs/1901.08755.

[65] S. Hardy, W. Henecka, H. Ivey-Law, R. Nock, G. Patrini, G. Smith, and B. Thorne, "Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption," *arXiv preprint arXiv:1711.10677*, 2017.

[66] Y. Liu, Y. Kang, X. Zhang, L. Li, Y. Cheng, T. Chen, M. Hong, and Q. Yang, "A communication efficient vertical federated learning framework," *CoRR*, vol. abs/1912.11187, 2019. [Online]. Available: http://arxiv.org/abs/1912.11187.

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

[68] Y. Hu, P. Liu, L. Kong, and D. Niu, "Learning privately over distributed features: An ADMM sharing approach," *arXiv preprint arXiv:1907.07735*, 2019.

[69] Y. Liu, Z. Yi, and T. Chen, "Backdoor attacks and defenses in feature-partitioned collaborative learning," *arXiv preprint arXiv:2007.03608*, 2020.

[70] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[71] H. Ludwig, N. Baracaldo, G. Thomas, Y. Zhou, A. Anwar, S. Rajamoni, Y. Ong, J. Radhakrishnan, A. Verma, M. Sinn, M. Purcell, A. Rawat, T. Minh, N. Holohan, S. Chakraborty, S. Whitherspoon, D. Steuer, L. Wynter, H. Hassan, S. Laguna, M. Yurochkin, M. Agarwal, E. Chuba, and A. Abay, "IBM federated learning: An enterprise framework white paper V0.1," *arXiv preprint arXiv:2007.10987*, 2020.

[72] J. Kang, Z. Xiong, D. Niyato, S. Xie, and J. Zhang, "Incentive mechanism for reliable federated learning: A joint optimization approach to combining reputation and contract theory," *IEEE Internet of Things Journal*, 2019.

[73] J. Kang, Z. Xiong, D. Niyato, H. Yu, Y.-C. Liang, and D. I. Kim, "Incentive design for efficient federated learning in mobile networks: A contract theory approach," in *IEEE VTS Asia Pacific Wireless Communications Symposium, APWCS 2019, Singapore, August 28–30, 2019*, pp. 1–5, 2019.

[74] H. Yu, Z. Liu, Y. Liu, T. Chen, M. Cong, X. Weng, D. Niyato, and Q. Yang, "A sustainable incentive scheme for federated learning," *IEEE Intelligent Systems*, vol. 35, no. 4, pp. 58–69, 2020.

[75] L. Lyu, J. Yu, K. Nandakumar, Y. Li, X. Ma, J. Jin, H. Yu, and K. S. Ng, "Towards fair and privacy-preserving federated deep models," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 11, pp. 2524–2541, 2020.

[76] Y. Kim, J. Sun, H. Yu, and X. Jiang, "Federated tensor factorization for computational phenotyping," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, August 13–17, 2017*, pp. 887–895, 2017. DOI: 10.1145/3097983.3098118.

[77] J. Ma, Q. Zhang, J. Lou, J. Ho, L. Xiong, and X. Jiang, "Privacy-preserving tensor factorization for collaborative health data analysis," in *ACM CIKM*, vol. 2, 2019.

[78] O. Gupta and R. Raskar, "Distributed learning of deep neural network over multiple agents," *Journal of Network and Computer Applications*, vol. 116, pp. 1–8, 2018.

[79] 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.

[80] I. Ceballos, V. Sharma, E. Mugica, A. Singh, A. Roman, P. Vepakomma, and R. Raskar, "SplitNN-driven vertical partitioning," *arXiv preprint arXiv:2008.04137*, 2018.

[81] A. Singh, P. Vepakomma, O. Gupta, and R. Raskar, "Detailed comparison of communication efficiency of split learning and federated learning," *arXiv preprint arXiv:1909.09145*, 2019.

[82] Z. Huo, B. Gu, and H. Huang, "Training neural networks using features replay," in *Advances in Neural Information Processing Systems*, pp. 6659–6668, 2018.

[83] M. Jaderberg, W. M. Czarnecki, S. Osindero, O. Vinyals, A. Graves, D. Silver, and K. Kavukcuoglu, "Decoupled neural interfaces using synthetic gradients," in *Proceedings of the 34th International Conference on Machine Learning – Volume 70*, JMLR.org, pp. 1627–1635, 2017.

[84] V. Sharma, P. Vepakomma, T. Swedish, K. Chang, J. Kalpathy-Cramer, and R. Raskar, "ExpertMatcher: Automating ML model selection for clients using hidden representations," *arXiv preprint arXiv:1910.03731*, 2019.

[85] P. Vepakomma, O. Singh, A. Gupta, and R. Raskar, "Nopeek: Information leakage reduction to share activations in distributed deep learning," *arXiv preprint arXiv:2008.09161*, 2020.

[86] G. J. Székely, M. L. Rizzo, and N. K. Bakirov, "Measuring and testing dependence by correlation of distances," *The Annals of Statistics*, vol. 35, no. 6, pp. 2769–2794, 2007.

[87] P. Vepakomma, C. Tonde, and A. Elgammal, "Supervised dimensionality reduction via distance correlation maximization," *Electronic Journal of Statistics*, vol. 12, no. 1, pp. 960–984, 2018.

[88] A. Singh, A. Chopra, V. Sharma, E. Garza, E. Zhang, P. Vepakomma, and R. Raskar, "DISCO: Dynamic and invariant sensitive channel obfuscation for deep neural networks," *arXiv preprint arXiv:2012.11025*, 2020.

[89] J. G. Moreno-Torres, T. Raeder, R. Alaiz-RodríGuez, N. V. Chawla, and F. Herrera, “A unifying view on dataset shift in classification,” *Pattern Recogn.*, vol. 45, no. 1, Jan. 2012.

[90] J. Quionero-Candela, M. Sugiyama, A. Schwaighofer, and N. D. Lawrence, *Dataset Shift in Machine Learning*. The MIT Press, 2009.

[91] K. Hsieh, A. Phanishayee, O. Mutlu, and P. B. Gibbons, *The non-IID data quagmire of decentralized machine learning*, 2019. [Online]. Available: https://arxiv.org/abs/1910.00189.

[92] H. Eichner, T. Koren, H. B. McMahan, N. Srebro, and K. Talwar, “Semi-cyclic stochastic gradient descent,” in *Accepted to ICML 2019*, 2019. [Online]. Available: https://arxiv.org/abs/1904.10120.

[93] T. Wang, J.-Y. Zhu, A. Torralba, and A. A. Efros, “Dataset distillation,” *arXiv preprint arXiv:1811.10959*, 2018.

[94] B. Woodworth, J. Wang, H. B. McMahan, and N. Srebro, “Graph oracle models, lower bounds, and gaps for parallel stochastic optimization,” in *Advances in Neural Information Processing Systems (NIPS)*, 2018. [Online]. Available: https://arxiv.org/abs/1805.10222.

[95] A. Cotter, O. Shamir, N. Srebro, and K. Sridharan, “Better mini-batch algorithms via accelerated gradient methods,” in *Advances in Neural Information Processing Systems*, J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Q. Weinberger, Eds., vol. 24, pp. 1647–1655, Curran Associates, Inc., 2011. [Online]. Available: https://proceedings.neurips.cc/paper/2011/file/b55ec28c52d5f6205684a473a2193564-Paper.pdf.

[96] O. Dekel, R. Gilad-Bachrach, O. Shamir, and L. Xiao, “Optimal distributed online prediction using mini-batches,” *J. Mach. Learn. Res.*, vol. 13, no. 6, pp. 165–202, 2012.

[97] G. Lan, “An optimal method for stochastic composite optimization,” *Math. Program.*, vol. 133, no. 1, pp. 365–397, 2012.

[98] H. Yu, S. Yang, and S. Zhu, “Parallel restarted SGD for non-convex optimization with faster convergence and less communication,” *arXiv preprint arXiv:1807.06629*, 2018.

[99] S. U. Stich, "Local SGD converges fast and communicates little," in *International Conference on Learning Representations (ICLR)*, 2019.

[100] S. U. Stich and S. P. Karimireddy, "The error-feedback framework: Better rates for SGD with delayed gradients and compressed communication," *arXiv:1909.05350*, 2019.

[101] 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*, PMLR, pp. 5132–5143, 2020.

[102] A. Khaled, K. Mishchenko, and P. Richtárik, *Better communication complexity for local SGD*, 2019. [Online]. Available: https://arxiv.org/abs/1909.04746.

[103] T. Lin, S. U. Stich, and M. Jaggi, "Don't use large mini-batches, use local SGD," *International Conference on Learning Representations (ICLR)*, 2020.

[104] K. K. Patel and A. Dieuleveut, "Communication trade-offs for synchronized distributed SGD with large step size," *NeurIPS*, 2019.

[105] A. Reisizadeh, A. Mokhtari, H. Hassani, A. Jadbabaie, and R. Pedarsani, "Fedpaq: A communication-efficient federated learning method with periodic averaging and quantization," *arXiv preprint arXiv:1909.13014*, 2019.

[106] B. Woodworth, K. K. Patel, S. U. Stich, Z. Dai, B. Bullins, H. B. McMahan, O. Shamir, and N. Srebro, "Is local SGD better than minibatch SGD?" *arXiv preprint arXiv:2002.07839*, 2020.

[107] S. Zhang, A. E. Choromanska, and Y. LeCun, "Deep learning with elastic averaging SGD," in *Advances in Neural Information Processing Systems*, pp. 685–693, 2015.

[108] F. Haddadpour, M. M. Kamani, M. Mahdavi, and V. R. Cadambe, "Local SGD with periodic averaging: Tighter analysis and adaptive synchronization," *arXiv preprint arXiv:1910.13598*, 2019.

[109] H. Karimi, J. Nutini, and M. Schmidt, "Linear convergence of gradient and proximal-gradient methods under the Polyak-łojasiewicz condition," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, Springer, pp. 795–811, 2016.

[110] J. Wang and G. Joshi, "Adaptive communication strategies for best error-runtime trade-offs in communication-efficient distributed SGD," in *Proceedings of the SysML Conference*, Apr. 2019. [Online]. Available: https://arxiv.org/abs/1810.08313.

[111] J. Wang, Q. Liu, H. Liang, G. Joshi, and H. V. Poor, "Tackling the objective inconsistency problem in heterogeneous federated optimization," in *Advances in Neural Information Processing Systems*, vol. 33, pp. 7611–7623, 2020.

[112] S. Dutta, G. Joshi, S. Ghosh, P. Dube, and P. Nagpurkar, "Slow and stale gradients can win the race: Error-runtime trade-offs in distributed SGD," *International Conference on Artificial Intelligence and Statistics (AISTATS)*, Apr. 2018. [Online]. Available: https://arxiv.org/abs/1803.01113.

[113] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, *Federated optimization in heterogeneous networks*, 2018. [Online]. Available: https://arxiv.org/abs/1812.06127.

[114] X. Li, W. Yang, S. Wang, and Z. Zhang, "Communication efficient decentralized training with multiple local updates," *arXiv preprint arXiv:1910.09126*, 2019.

[115] A. Khaled, K. Mishchenko, and P. Richtárik, *First analysis of local GD on heterogeneous data*, 2019. [Online]. Available: https://arxiv.org/abs/1909.04715.

[116] 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.

[117] H. Yu, R. Jin, and S. Yang, "On the linear speedup analysis of communication efficient momentum SGD for distributed nonconvex optimization," *arXiv preprint arXiv:1905.03817*, 2019.

[118] C. Xie, O. Koyejo, I. Gupta, and H. Lin, "Local adaalter: Communication-efficient stochastic gradient descent with adaptive learning rates," *arXiv preprint arXiv:1911.09030*, 2019.

[119] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *J. Mach. Learn. Res.*, vol. 12, no. 61, pp. 2121–2159, 2011.

[120] H. B. McMahan and M. Streeter, "Adaptive bound optimization for online convex optimization," *arXiv preprint arXiv:1002.4908*, 2010.

[121] 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.

[122] T.-M. H. Hsu, H. Qi, and M. Brown, "Measuring the effects of non-identical data distribution for federated visual classification," *arXiv preprint arXiv:1909.06335*, 2019.

[123] J. Wang, V. Tantia, N. Ballas, and M. Rabbat, "SlowMo: Improving communication-efficient distributed SGD with slow momentum," *arXiv preprint arXiv:1910.00643*, 2019.

[124] 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.

[125] I. Almeida and J. Xavier, "DJAM: Distributed Jacobi asynchronous method for learning personal models," *IEEE Signal Process. Lett.*, vol. 25, no. 9, pp. 1389–1392, 2018.

[126] K. Wang, R. Mathews, C. Kiddon, H. Eichner, F. Beaufays, and D. Ramage, "Federated evaluation of on-device personalization," *arXiv preprint arXiv:1910.10252*, 2019.

[127] Y. Zhang and Q. Yang, "A survey on multi-task learning," *CoRR*, vol. abs/1707.08114, 2017. [Online]. Available: http://arxiv.org/abs/1707.08114.

[128] V. Smith, C.-K. Chiang, M. Sanjabi, and A. S. Talwalkar, "Federated multi-task learning," in *NIPS*, 2017.

[129] Y. Mansour, M. Mohri, J. Ro, and A. T. Suresh, "Three approaches for personalization with applications to federated learning," *arXiv preprint arXiv:2002.10619*, 2020.

[130] S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. W. Vaughan, "A theory of learning from different domains," *Mach. Learn.*, vol. 79, no. 1–2, pp. 151–175, 2010.

[131] C. Cortes and M. Mohri, "Domain adaptation and sample bias correction theory and algorithm for regression," *Theor. Comput. Sci.*, vol. 519, pp. 103–126, 2014.

[132] C. Cortes, M. Mohri, A. T. Suresh, and N. Zhang, "Multiple-source adaptation with domain classifiers," *arXiv preprint arXiv: 2008.11036*, 2020.

[133] Y. Mansour, M. Mohri, and A. Rostamizadeh, "Domain adaptation: Learning bounds and algorithms," *arXiv preprint arXiv: 0902.3430*, 2009.

[134] Y. Mansour, M. Mohri, A. T. Suresh, and K. Wu, "A theory of multiple-source adaptation with limited target labeled data," *arXiv preprint arXiv:2007.09762*, 2020.

[135] J. Baxter, "A model of inductive bias learning," *J. Artif. Intell. Res.*, vol. 12, pp. 149–198, 2000.

[136] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proceedings of the 34th International Conference on Machine Learning*, 2017.

[137] A. Nichol, J. Achiam, and J. Schulman, "On first-order meta-learning algorithms," *arXiv preprint arXiv:1803.02999*, 2018.

[138] M. Khodak, M.-F. Balcan, and A. Talwalkar, "Adaptive gradient-based meta-learning methods," in *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[139] Y. Jiang, J. Konečný, K. Rush, and S. Kannan, "Improving federated learning personalization via model agnostic meta learning," *arXiv preprint arXiv:1909.12488*, 2019.

[140] K. C. Sim, F. Beaufays, A. Benard, D. Guliani, A. Kabel, N. Khare, T. Lucassen, P. Zadrazil, H. Zhang, L. Johnson, G. Motta, and L. Zhou, "Personalization of end-to-end speech recognition on mobile devices for named entities," *arXiv preprint arXiv:1912.09251*, 2019.

[141] A. Fallah, A. Mokhtari, and A. Ozdaglar, "Personalized federated learning: A meta-learning approach," *arXiv preprint arXiv:2002.07948*, 2020.

[142] J. Li, M. Khodak, S. Caldas, and A. Talwalkar, "Differentially private meta-learning," *arXiv preprint arXiv:1909.05830*, 2019.

[143] B. M. Lake, R. Salakhutdinov, J. Gross, and J. B. Tenenbaum, "One shot learning of simple visual concepts," in *Proceedings of the Conference of the Cognitive Science Society (CogSci)*, 2017.

[144] S. Ravi and H. Larochelle, "Optimization as a model for few-shot learning," in *Proceedings of the 5th International Conference on Learning Representations*, 2017.

[145] D. L. Silver, Q. Yang, and L. Li, "Lifelong machine learning systems: Beyond learning algorithms," in *AAAI Spring Symposium Series*, 2013.

[146] J. Snell, K. Swersky, and R. S. Zemel, "Prototypical networks for few-shot learning," in *Advances in Neural Information Processing Systems*, pp. 4080–4090, 2017.

[147] J. Hoffman, M. Mohri, and N. Zhang, "Algorithms and theory for multiple-source adaptation," in *Advances in Neural Information Processing Systems*, pp. 8246–8256, 2018.

[148] Y. Mansour, M. Mohri, and A. Rostamizadeh, "Domain adaptation with multiple sources," in *Advances in Neural Information Processing Systems*, pp. 1041–1048, 2009.

[149] M. Mohri, G. Sivek, and A. T. Suresh, "Agnostic Federated Learning," in *ICML*, 2019.

[150] P. Christen, *Data Matching: Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection*. Springer Science & Business Media, 2012.

[151] R. Schnell, T. Bachteler, and J. Reiher, "A novel error-tolerant anonymous linking code,"

[152] R. Schnell, "Efficient private record linkage of very large datasets," in *59th World Statistics Congress*, 2013.

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

[154] G. Patrini, R. Nock, S. Hardy, and T. S. Caetano, "Fast learning from distributed datasets without entity matching," in *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9–15 July 2016*, pp. 1909–1917, 2016. [Online]. Available: http://www.ijcai.org/Abstract/16/273.

[155] R. D. King, C. Feng, and A. Sutherland, “StatLog: Comparison of classification algorithms on large real-world problems,” *Appl. Artif. Intell.*, vol. 9, no. 3, pp. 289–333, 1995.
[156] R. Kohavi and G. H. John, “Automatic parameter selection by minimizing estimated error,” in *Machine Learning Proceedings 1995*, Elsevier, 1995, pp. 304–312.
[157] B. D. Ripley, “Statistical aspects of neural networks,” *Networks and Chaos—Statistical and Probabilistic Aspects*, vol. 50, Chapman and Hall, 1993, pp. 40–123.
[158] J. S. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, “Algorithms for hyper-parameter optimization,” in *Advances in Neural Information Processing Systems*, pp. 2546–2554, 2011.
[159] S. Falkner, A. Klein, and F. Hutter, “BOHB: Robust and efficient hyperparameter optimization at scale,” *arXiv preprint arXiv:1807.01774*, 2018.
[160] F. Pedregosa, “Hyperparameter optimization with approximate gradient,” *arXiv preprint arXiv:1602.02355*, 2016.
[161] J. Snoek, O. Rippel, K. Swersky, R. Kiros, N. Satish, N. Sundaram, M. Patwary, M. Prabhat, and R. Adams, “Scalable Bayesian optimization using deep neural networks,” in *International Conference on Machine Learning*, pp. 2171–2180, 2015.
[162] Z. Charles and J. Konečnỳ, “On the outsized importance of learning rates in local update methods,” *arXiv preprint arXiv:2007.00878*, 2020.
[163] K. A. Bonawitz, F. Salehi, J. Konečný, B. McMahan, and M. Gruteser, “Federated learning with autotuned communication-efficient secure aggregation,” in *2019 53rd Asilomar Conference on Signals, Systems, and Computers*, IEEE, 2019.
[164] O. Thakkar, G. Andrew, and H. B. McMahan, “Differentially private learning with adaptive clipping,” *arXiv preprint arXiv:1905.03871*, 2019.
[165] I. Bello, B. Zoph, V. Vasudevan, and Q. V. Le, “Neural optimizer search with reinforcement learning,” in *Proceedings of the 34th International Conference on Machine Learning – Volume 70*, JMLR.org, pp. 459–468, 2017.

[166] T. Elsken, J. Hendrik Metzen, and F. Hutter, “Efficient multi-objective neural architecture search via Lamarckian evolution,” *arXiv preprint arXiv:1804.09081*, 2018.

[167] H. Liu, K. Simonyan, and Y. Yang, “DARTS: Differentiable architecture search,” *arXiv preprint arXiv:1806.09055*, 2018.

[168] R. Luo, F. Tian, T. Qin, E. Chen, and T.-Y. Liu, “Neural architecture optimization,” in *Advances in Neural Information Processing Systems*, pp. 7816–7827, 2018.

[169] C. He, H. Ye, L. Shen, and T. Zhang, “Milenas: Efficient neural architecture search via mixed-level reformulation,” in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[170] E. Real, S. Moore, A. Selle, S. Saxena, Y. L. Suematsu, J. Tan, Q. V. Le, and A. Kurakin, “Large-scale evolution of image classifiers,” in *Proceedings of the 34th International Conference on Machine Learning – Volume 70*, JMLR.org, pp. 2902–2911, 2017.

[171] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, “Regularized evolution for image classifier architecture search,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 4780–4789, 2019.

[172] H. Pham, M. Guan, B. Zoph, Q. Le, and J. Dean, “Efficient neural architecture search via parameter sharing,” in *International Conference on Machine Learning*, pp. 4092–4101, 2018.

[173] S. Xie, H. Zheng, C. Liu, and L. Lin, “SNAS: Stochastic neural architecture search,” *arXiv preprint arXiv:1812.09926*, 2018.

[174] A. Gaier and D. Ha, “Weight agnostic neural networks,” *arXiv preprint arXiv:1906.04358*, 2019.

[175] C. He, M. Annavaram, and S. Avestimehr, “FedNAS: Federated deep learning via neural architecture search,” *arXiv preprint arXiv:2004.08546*, 2020.

[176] S. Augenstein, H. B. McMahan, D. Ramage, S. Ramaswamy, P. Kairouz, M. Chen, R. Mathews, and B. A. y Arcas, *Generative models for effective ML on private, decentralized datasets*, 2019. [Online]. Available: https://arxiv.org/abs/1911.06679.

[177] J. Konečný, 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.

[178] J. Acharya, C. L. Canonne, and H. Tyagi, "Inference under information constraints I: Lower bounds from chi-square contraction," *IEEE Trans. Inform. Theor.*, vol. 66, no. 12, pp. 7835–7855, 2020.

[179] L. P. Barnes, Y. Han, and A. Ozgur, "Lower bounds for learning distributions under communication constraints via Fisher information," *J. Mach. Learn. Res.*, vol. 21, no. 236, pp. 1–30, 2020.

[180] M. Braverman, A. Garg, T. Ma, H. L. Nguyen, and D. P. Woodruff, "Communication lower bounds for statistical estimation problems via a distributed data processing inequality," in *Proceedings of the Forty-Eighth Annual ACM Symposium on Theory of Computing*, pp. 1011–1020, ACM, 2016.

[181] L. P. Barnes, H. A. Inan, B. Isik, and A. Ozgur, "rTop-k: A statistical estimation approach to distributed SGD," *arXiv preprint arXiv:2005.10761*, 2020.

[182] Y. Zhang, J. Duchi, M. I. Jordan, and M. J. Wainwright, "Information-theoretic lower bounds for distributed statistical estimation with communication constraints," in *Advances in Neural Information Processing Systems*, pp. 2328–2336, 2013.

[183] Y. Han, A. Özgür, and T. Weissman, "Geometric lower bounds for distributed parameter estimation under communication constraints," in *Proceedings of Machine Learning Research*, pp. 1–26, 75, 2018.

[184] D. Alistarh, D. Grubic, J. Li, R. Tomioka, and M. Vojnovic, "QSGD: Communication-efficient SGD via gradient quantization and encoding," in *NIPS – Advances in Neural Information Processing Systems*, pp. 1709–1720, 2017.

[185] D. Basu, D. Data, C. Karakus, and S. N. Diggavi, "Qsparse-local-SGD: Distributed SGD with quantization, sparsification, and local computations," *IEEE Sel. Areas Inf. Theor.*, vol. 1, no. 1, pp. 217–226, 2020.

[186] S. Horvath, C.-Y. Ho, L. Horvath, A. N. Sahu, M. Canini, and P. Richtarik, "Natural compression for distributed deep learning," *arXiv preprint arXiv:1905.10988*, 2019.

[187] J. Konečný and P. Richtárik, "Randomized distributed mean estimation: Accuracy vs. communication," *Frontiers in Applied Mathematics and Statistics*, vol. 4, p. 62, 2018.

[188] A. T. Suresh, F. X. Yu, S. Kumar, and H. B. McMahan, "Distributed mean estimation with limited communication," in *Proceedings of the 34th International Conference on Machine Learning – Volume 70*, JMLR.org, pp. 3329–3337, 2017.

[189] S. Chraibi, A. Khaled, D. Kovalev, P. Richtárik, A. Salim, and M. Takáč, "Distributed fixed point methods with compressed iterates," *arXiv preprint arXiv:1912.09925*, 2019.

[190] S. Caldas, J. Konečný, H. B. McMahan, and A. Talwalkar, "Expanding the reach of federated learning by reducing client resource requirements," *arXiv preprint arXiv:1812.07210*, 2018.

[191] J. Hamer, M. Mohri, and A. T. Suresh, "Fedboost: A communication-efficient algorithm for federated learning," in *International Conference on Machine Learning*, PMLR, pp. 3973–3983, 2020.

[192] C. He, M. Annavaram, and S. Avestimehr, "Group knowledge transfer: Federated learning of large cnns at the edge," in *Advances in Neural Information Processing Systems*, vol. 34, pp. 14068–14080, 2020.

[193] S. P. Karimireddy, Q. Rebjock, S. Stich, and M. Jaggi, "Error feedback fixes SignSGD and other gradient compression schemes," in *ICML*, 2019.

[194] Y. Lin, S. Han, H. Mao, Y. Wang, and W. J. Dally, "Deep gradient compression: Reducing the communication bandwidth for distributed training," *arXiv preprint arXiv:1712.01887*, 2017.

[195] F. Sattler, S. Wiedemann, K.-R. Müller, and W. Samek, "Robust and communication-efficient federated learning from non-IID data," *arXiv preprint arXiv:1903.02891*, 2019.

[196] T. Vogels, S. P. Karimireddy, and M. Jaggi, "PowerSGD: Practical low-rank gradient compression for distributed optimization," in *NeurIPS 2019 – Advances in Neural Information Processing Systems*, vol. 32, 2019.

[197] D. Blalock, J. J. G. Ortiz, J. Frankle, and J. Guttag, "What is the state of neural network pruning?" *arXiv preprint arXiv:2003.03033*, 2020.

[198] M. Courbariaux, Y. Bengio, and J.-P. David, "BinaryConnect: Training deep neural networks with binary weights during propagations," in *Advances in Neural Information Processing Systems*, pp. 3123–3131, 2015.

[199] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.

[200] D. Lin, S. Talathi, and S. Annapureddy, "Fixed point quantization of deep convolutional networks," in *International Conference on Machine Learning*, pp. 2849–2858, 2016.

[201] D. Oktay, J. Ballé, S. Singh, and A. Shrivastava, "Model compression by entropy penalized reparameterization," *arXiv preprint arXiv:1906.06624*, 2019.

[202] M. Zhu and S. Gupta, "To prune, or not to prune: Exploring the efficacy of pruning for model compression," *arXiv preprint arXiv:1710.01878*, 2017.

[203] T. M. Cover and J. A. Thomas, *Elements of Information Theory.* John Wiley & Sons, 2012.

[204] X. Wu, R. Guo, A. T. Suresh, S. Kumar, D. N. Holtmann-Rice, D. Simcha, and F. X. Yu, "Multiscale quantization for fast similarity search," in *Advances in Neural Information Processing Systems*, pp. 5745–5755, 2017.

[205] V. Gandikota, R. K. Maity, and A. Mazumdar, "VqSGD: Vector quantized stochastic gradient descent," *arXiv preprint arXiv:1911.07971*, 2019.

[206] 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 federated learning on user-held data," *arXiv preprint arXiv:1611.04482*, 2016.

[207] 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*, ACM, pp. 308–318, 2016.

[208] J. H. Bell, K. A. Bonawitz, A. Gascón, T. Lepoint, and M. Raykova, "Secure single-server aggregation with (poly)logarithmic overhead," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, ACM, pp. 1253–1269, 2020.

[209] N. Agarwal, A. T. Suresh, F. X. Yu, S. Kumar, and B. McMahan, "CpSGD: Communication-efficient and differentially-private distributed SGD," in *Advances in Neural Information Processing Systems*, pp. 7564–7575, 2018.

[210] E. Jeong, S. Oh, H. Kim, J. Park, M. Bennis, and S.-L. Kim, "Communication-efficient on-device machine learning: Federated distillation and augmentation under non-IID private data," *CoRR*, vol. abs/1811.11479, 2018. [Online]. Available: http://arxiv.org/abs/1811.11479.

[211] J. Park, S. Samarakoon, M. Bennis, and M. Debbah, "Wireless network intelligence at the edge," *CoRR*, vol. abs/1812.02858, 2018. [Online]. Available: http://arxiv.org/abs/1812.02858.

[212] S. Samarakoon, M. Bennis, W. Saad, and M. Debbah, "Federated learning for ultra-reliable low-latency V2V communications," *CoRR*, vol. abs/1805.09253, 2018. [Online]. Available: http://arxiv.org/abs/1805.09253.

[213] O. Abari, H. Rahul, and D. Katabi, "Over-the-air function computation in sensor networks," *CoRR*, vol. abs/1612.02307, 2016. [Online]. Available: http://arxiv.org/abs/1612.02307.

[214] A. Elgabli, J. Park, C. B. Issaid, and M. Bennis, "Harnessing wireless channels for scalable and privacy-preserving federated learning," *IEEE Trans. Comm.*, to be published.

[215] Y. Zhao, C. Yu, P. Zhao, and J. Liu, "Decentralized online learning: Take benefits from others' data without sharing your own to track global trend," *arXiv preprint arXiv:1901.10593*, 2019.

[216] K. Shridhar, F. Laumann, and M. Liwicki, "A comprehensive guide to Bayesian convolutional neural network with variational inference," *arXiv preprint arXiv:1901.02731*, 2019.

[217] A. Lalitha, X. Wang, O. Kilinc, Y. Lu, T. Javidi, and F. Koushanfar, "Decentralized Bayesian learning over graphs," *arXiv preprint: arXiv:1905.10466*, 2019.

[218] F. Mireshghallah, M. Taram, P. Vepakomma, A. Singh, R. Raskar, and E. Hadi, "Privacy in deep learning: A survey," *arXiv preprint arXiv:2004.12254*, 2020.

[219] A. Bittau, Ú. Erlingsson, P. Maniatis, I. Mironov, A. Raghunathan, D. Lie, M. Rudominer, U. Kode, J. Tinnes, and B. Seefeld, "Prochlo: Strong privacy for analytics in the crowd," in *Proceedings of the 26th Symposium on Operating Systems Principles,* SOSP '17, pp. 441–459, New York, NY, USA: ACM, 2017. DOI: 10.1145/3132747.3132769.

[220] NSA, *Defense in depth: A practical strategy for achieving Information Assurance in today's highly networked environments*, 2012. [Online]. Available: https://apps.nsa.gov/iaarchive/library/ia-guidance/archive/defense-in-depth.cfm.

[221] C.-C. Y. Andrew, "How to generate and exchange secrets (extended abstract)," in *FOCS*, pp. 162–167, IEEE Computer Society, 1986.

[222] T. Araki, J. Furukawa, Y. Lindell, A. Nof, and K. Ohara, "High-throughput semi-honest secure three-party computation with an honest majority," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ACM, pp. 805–817, 2016.

[223] P. Bogetoft, D. L. Christensen, I. Damgård, M. Geisler, T. P. Jakobsen, M. Krøigaard, J. D. Nielsen, J. B. Nielsen, K. Nielsen, J. Pagter, M. I. Schwartzbach, and T. Toft, "Secure multiparty computation goes live," in *Financial Cryptography,* Lecture Notes in Computer Science, vol. 5628, pp. 325–343, Springer, 2009.

[224] D. Bogdanov, R. Talviste, and J. Willemson, "Deploying secure multi-party computation for financial data analysis – (short paper)," in *Financial Cryptography,* Lecture Notes in Computer Science, vol. 7397, pp. 57–64, Springer, 2012.

[225] A. Lapets, N. Volgushev, A. Bestavros, F. Jansen, and M. Varia, "Secure MPC for analytics as a web application," in *SecDev*, pp. 73–74, IEEE Computer Society, 2016.

[226] J. Furukawa, Y. Lindell, A. Nof, and O. Weinstein, "High-throughput secure three-party computation for malicious adversaries and an honest majority," in *EUROCRYPT (2),* Lecture Notes in Computer Science, vol. 10211, pp. 225–255, 2017.

[227] M. Ion, B. Kreuter, E. Nergiz, S. Patel, S. Saxena, K. Seth, D. Shanahan, and M. Yung, "Private intersection-sum protocol with applications to attributing aggregate ad conversions," *IACR Cryptology ePrint Archive*, vol. 2017, p. 738, 2017.

[228] M. Ion, B. Kreuter, A. E. Nergiz, S. Patel, M. Raykova, S. Saxena, K. Seth, D. Shanahan, and M. Yung, "On deploying secure computing commercially: Private intersection-sum protocols and their business applications," *IACR Cryptology ePrint Archive*, vol. 2019, p. 723, 2019.

[229] Y. Ishai, J. Kilian, K. Nissim, and E. Petrank, "Extending oblivious transfers efficiently," in *CRYPTO,* Lecture Notes in Computer Science, vol. 2729, pp. 145–161, Springer, 2003.

[230] F. Bourse, M. Minelli, M. Minihold, and P. Paillier, "Fast homomorphic evaluation of deep discretized neural networks," in *CRYPTO (3),* Lecture Notes in Computer Science, vol. 10993, pp. 483–512, Springer, 2018.

[231] R. Gilad-Bachrach, N. Dowlin, K. Laine, K. E. Lauter, M. Naehrig, and J. Wernsing, "CryptoNets: Applying neural networks to encrypted data with high throughput and accuracy," in *Proceedings of the 33rd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19–24, 2016*, pp. 201–210, 2016. [Online]. Available: http://proceedings.mlr.press/v48/gilad-bachrach16.html.

[232] A. Gascón, P. Schoppmann, B. Balle, M. Raykova, J. Doerner, S. Zahur, and D. Evans, "Privacy-preserving distributed linear regression on high-dimensional data," *PoPETs*, vol. 2017, no. 4, pp. 345–364, 2017.

[233] N. Agrawal, A. S. Shamsabadi, M. J. Kusner, and A. Gascón, “QUOTIENT: Two-party secure neural network training and prediction,” in *Proceedings of the ACM Conference on Computer and Communication Security (CCS)*, 2019.

[234] O. Goldreich, S. Micali, and A. Wigderson, “How to play any mental game,” in *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing,* STOC ’87, pp. 218–229, New York, New York, USA: ACM, 1987. DOI: 10.1145/28395.28420.

[235] V. Nikolaenko, U. Weinsberg, S. Ioannidis, M. Joye, D. Boneh, and N. Taft, “Privacy-preserving ridge regression on hundreds of millions of records,” in *IEEE Symposium on Security and Privacy*, pp. 334–348, IEEE Computer Society, 2013.

[236] P. Mohassel and Y. Zhang, “SecureML: A system for scalable privacy-preserving machine learning,” in *IEEE Symposium on Security and Privacy*, pp. 19–38, IEEE Computer Society, 2017.

[237] A. Barak, D. Escudero, A. P. K. Dalskov, and M. Keller, “Secure evaluation of quantized neural networks,” *IACR Cryptology ePrint Archive*, vol. 2019, p. 131, 2019. [Online]. Available: https://eprint.iacr.org/2019/131.

[238] C. Gentry, “Fully homomorphic encryption using ideal lattices,” in *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing*, pp. 169–178, 2009.

[239] Z. Brakerski, “Fully homomorphic encryption without modulus switching from classical GapSVP,” in *CRYPTO,* Lecture Notes in Computer Science, vol. 7417, pp. 868–886, Springer, 2012.

[240] J. Fan and F. Vercauteren, “Somewhat practical fully homomorphic encryption,” *IACR Cryptology ePrint Archive*, vol. 2012, p. 144, 2012.

[241] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, “(leveled) Fully homomorphic encryption without bootstrapping,” in *ITCS*, pp. 309–325, ACM, 2012.

[242] J.-S. Coron, T. Lepoint, and M. Tibouchi, “Scale-invariant fully homomorphic encryption over the integers,” in *Public Key Cryptography,* Lecture Notes in Computer Science, vol. 8383, pp. 311–328, Springer, 2014.

[243] *HElib*, https://github.com/homenc/HElib, Oct. 2019.

[244] *Lattigo 2.0.0*, [Online]: http://github.com/ldsec/lattigo, Oct. 2020, EPFL-LDS.

[245] *PALISADE lattice cryptography library*, https://gitlab.com/palisade/palisade-release, Oct. 2019.

[246] *Microsoft SEAL (release 3.6)*, https://github.com/Microsoft/SEAL, Nov. 2020, Microsoft Research, Redmond, WA.

[247] *SHELL*, https://github.com/google/shell-encryption, Dec. 2020. Google.

[248] S. S. Sathya, P. Vepakomma, R. Raskar, R. Ramachandra, and S. Bhattacharya, "A review of homomorphic encryption libraries for secure computation," *arXiv preprint arXiv:1812.02428*, 2018.

[249] M. Chenal and Q. Tang, "On key recovery attacks against existing somewhat homomorphic encryption schemes," in *LATINCRYPT,* Lecture Notes in Computer Science, vol. 8895, pp. 239–258, Springer, 2014.

[250] L. Reyzin, A. D. Smith, and S. Yakoubov, "Turning HATE into LOVE: Homomorphic ad hoc threshold encryption for scalable MPC," *IACR Cryptology ePrint Archive*, vol. 2018, p. 997, 2018.

[251] E. Roth, D. Noble, B. H. Falk, and A. Haeberlen, "Honeycrisp: Large-scale differentially private aggregation without a trusted core," in *SOSP*, pp. 196–210, ACM, 2019.

[252] P. Subramanyan, R. Sinha, I. Lebedev, S. Devadas, and S. A. Seshia, "A formal foundation for secure remote execution of enclaves," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, ACM, pp. 2435–2450, 2017.

[253] R. Intel, "Architecture instruction set extensions programming reference," *Intel Corporation,* Feb., 2012.

[254] V. Costan and S. Devadas, "Intel SGX explained," *IACR Cryptology ePrint Archive*, vol. 2016, no. 086, pp. 1–118, 2016.

[255] Arm trustzone, *Arm TrustZone Technology*, https://developer.arm.com/ip-products/security-ip/trustzone (accessed Dec. 5, 2019).

[256] Android trusty, *Android trusty TEE*, https://source.android.com/security/trusty (accessed Dec. 5, 2019).

[257] V. Costan, I. Lebedev, and S. Devadas, “Sanctum: Minimal hardware extensions for strong software isolation,” in *25th USENIX Security Symposium (USENIX Security 16)*, pp. 857–874, 2016.

[258] F. Tramèr and D. Boneh, “Slalom: Fast, verifiable and private execution of neural networks in trusted hardware,” in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=rJVorjCcKQ.

[259] J. Van Bulck, M. Minkin, O. Weisse, D. Genkin, B. Kasikci, F. Piessens, M. Silberstein, T. F. Wenisch, Y. Yarom, and R. Strackx, “Foreshadow: Extracting the keys to the intel SGX kingdom with transient out-of-order execution,” in *27th USENIX Security Symposium (USENIX Security 18)*, pp. 991–1008, 2018.

[260] G. Ács and C. Castelluccia, “I have a DREAM!: DiffeRentially privatE smArt Metering,” in *Proceedings of the 13th International Conference on Information Hiding,* IH’11, pp. 118–132, Berlin, Heidelberg: Springer-Verlag, 2011. [Online]. Available: http://dl.acm.org/citation.cfm?id=2042445.2042457.

[261] J. So, B. Guler, and A. S. Avestimehr, “Turbo-aggregate: Breaking the quadratic aggregation barrier in secure federated learning,” *arXiv preprint arXiv:2002.04156*, 2020.

[262] S. Goryczka and L. Xiong, “A comprehensive comparison of multiparty secure additions with differential privacy,” *IEEE Trans. Dependable Sec. Comput.*, vol. 14, no. 5, pp. 463–477, 2017.

[263] T.-H. H. Chan, E. Shi, and D. Song, “Privacy-preserving stream aggregation with fault tolerance,” in *International Conference on Financial Cryptography and Data Security*, Springer, pp. 200–214, 2012.

[264] S. Halevi, Y. Lindell, and B. Pinkas, “Secure computation on the web: Computing without simultaneous interaction,” in *Annual Cryptology Conference*, Springer, pp. 132–150, 2011.

[265] E. Shi, H. T. H. Chan, E. Rieffel, R. Chow, and D. Song, “Privacy-preserving aggregation of time-series data,” in *Annual Network & Distributed System Security Symposium (NDSS)*, 2011.

[266] M. Burkhart, M. Strasser, D. Many, and X. Dimitropoulos, "SEPIA: Privacy-preserving aggregation of multi-domain network events and statistics," *Network*, vol. 1, no. 101101, 2010.

[267] H. Corrigan-Gibbs and D. Boneh, "Prio: Private, robust, and scalable computation of aggregate statistics," in *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*, pp. 259–282, 2017.

[268] D. Lie and P. Maniatis, "Glimmers: Resolving the privacy/trust quagmire," in *Proceedings of the 16th Workshop on Hot Topics in Operating Systems*, ACM, pp. 94–99, 2017.

[269] D. Chaum, "Untraceable electronic mail, return addresses, and digital pseudonyms," *Communications of the ACM*, vol. 24, no. 2, 1981.

[270] A. Kwon, D. Lazar, S. Devadas, and B. Ford, "Riffle," *Proceedings on Privacy Enhancing Technologies*, vol. 2016, no. 2, pp. 115–134, 2016.

[271] R. Dingledine, N. Mathewson, and P. Syverson, *Tor: The second-generation onion router*, Tech. Rep., Naval Research Lab, Washington, DC, 2004.

[272] E. Kushilevitz and R. Ostrovsky, "Replication is not needed: Single database, computationally-private information retrieval," in *Proc. of the 38th Annu. IEEE Symp. on Foundations of Computer Science*, pp. 364–373, 1997.

[273] B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan, "Private information retrieval," *J. ACM*, vol. 45, no. 6, pp. 965–981, Nov. 1998.

[274] R. Sion and B. Carbunar, "On the computational practicality of private information retrieval," in *Proceedings of the Network and Distributed Systems Security Symposium*, Internet Society, pp. 2006–06, 2007.

[275] C. Aguilar-Melchor and P. Gaborit, "A lattice-based computationally-efficient private information retrieval protocol," *Cryptol. ePrint Arch., Report*, vol. 446, 2007.

[276] C. Aguilar-Melchor, J. Barrier, L. Fousse, and M.-O. Killijian, "XPIR: Private information retrieval for everyone," *Proceedings on Privacy Enhancing Technologies*, vol. 2016, no. 2, pp. 155–174, 2016.

[277] S. Angel, H. Chen, K. Laine, and S. T. V. Setty, "PIR with compressed queries and amortized query processing," in *IEEE Symposium on Security and Privacy*, pp. 962–979, IEEE Computer Society, 2018.

[278] C. Gentry and S. Halevi, "Compressible FHE with applications to PIR," in *TCC (2),* Lecture Notes in Computer Science, vol. 11892, pp. 438–464, Springer, 2019.

[279] F. Olumofin and I. Goldberg, "Revisiting the computational practicality of private information retrieval," in *International Conference on Financial Cryptography and Data Security*, Springer, pp. 158–172, 2011.

[280] A. Ali, T. Lepoint, S. Patel, M. Raykova, P. Schoppmann, K. Seth, and K. Yeo, "Communication-computation trade-offs in PIR," *IACR Cryptol. ePrint Arch.*, vol. 2019, p. 1483, 2019.

[281] S. Kadhe, B. Garcia, A. Heidarzadeh, S. E. Rouayheb, and A. Sprintson, "Private information retrieval with side information: The single server case," in *2017 55th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 1099–1106, Oct. 2017. DOI: 10.1109/ALLERTON.2017.8262860.

[282] S. Patel, G. Persiano, and K. Yeo, "Private stateful information retrieval," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security,* CCS '18, pp. 1002–1019, New York, NY, USA: ACM, 2018. DOI: 10.1145/3243734.3243821.

[283] H. Corrigan-Gibbs and D. Kogan, "Private information retrieval with sublinear online time," *IACR Cryptology ePrint Archive*, vol. 2019, p. 1075, 2019.

[284] D. Woodruff and S. Yekhanin, "A geometric approach to information-theoretic private information retrieval," in *20th Annual IEEE Conference on Computational Complexity (CCC'05)*, pp. 275–284, Jun. 2005. DOI: 10.1109/CCC.2005.2.

[285] R. G. L. D'Oliveira and S. E. Rouayheb, "Lifting private information retrieval from two to any number of messages," in *2018 IEEE International Symposium on Information Theory (ISIT)*, pp. 1744–1748, Jun. 2018. DOI: 10.1109/ISIT.2018.8437805.

[286] R. Bitar and S. E. Rouayheb, "Staircase-PIR: Universally robust private information retrieval," in *2018 IEEE Information Theory Workshop (ITW)*, pp. 1–5, Nov. 2018. DOI: 10.1109/ITW.2018.8613532.

[287] T. Lepoint, S. Patel, M. Raykova, K. Seth, and N. Trieu, "Private join and compute from PIR with default," *IACR Cryptol. ePrint Arch.*, vol. 2020, p. 1011, 2020.

[288] C. Naim, F. Ye, and S. E. Rouayheb, "ON-OFF privacy with correlated requests," in *2019 IEEE International Symposium on Information Theory (ISIT)*, Jul. 2019.

[289] F. Ye, C. Naim, and S. El Rouayheb, "Preserving ON-OFF privacy for past and future requests," in *2019 IEEE Information Theory Workshop (ITW)*, Aug. 2019.

[290] C. Dwork, F. McSherry, K. Nissim, and A. D. Smith, "Calibrating noise to sensitivity in private data analysis," in *IACR Theory of Cryptography Conference (TCC), New York,* Lecture Notes in Computer Science, vol. 3876, pp. 265–284, Springer-Verlag, 2006. DOI: 10.1007/11681878_14.

[291] C. Dwork, "Differential privacy: A survey of results," in *International Conference on Theory and Applications of Models of Computation*, Springer, pp. 1–19, 2008.

[292] C. Dwork and A. Roth, "The algorithmic foundations of differential privacy," *Foundations and Trends in Theoretical Computer Science*, vol. 9, no. 3–4, pp. 211–407, 2014.

[293] S. L. Warner, "Randomized response: A survey technique for eliminating evasive answer bias," *J. Am. Stat. Assoc.*, vol. 60, no. 309, pp. 63–69, 1965.

[294] S. P. Kasiviswanathan, H. K. Lee, K. Nissim, S. Raskhodnikova, and A. D. Smith, "What can we learn privately?" *SIAM J. Comput.*, vol. 40, no. 3, pp. 793–826, 2011.

[295] B. Ding, J. Kulkarni, and S. Yekhanin, "Collecting telemetry data privately," in *Advances in Neural Information Processing Systems,* vol. 30, Dec. 2017. [Online]. Available: https://www.microsoft.com/en-us/research/publication/collecting-telemetry-data-privately/.

[296] V. Pihur, A. Korolova, F. Liu, S. Sankuratripati, M. Yung, D. Huang, and R. Zeng, "Differentially-private 'draw and discard' machine learning," *CoRR*, vol. abs/1807.04369, 2018. [Online]. Available: http://arxiv.org/abs/1807.04369.

[297] J. Ullman, *Tight lower bounds for locally differentially private selection*, Tech. Rep., abs/1802.02638, 2018. [Online]. Available: http://arxiv.org/abs/1802.02638.

[298] A. Cheu, A. Smith, J. Ullman, D. Zeber, and M. Zhilyaev, "Distributed differential privacy via shuffling," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, Springer, pp. 375–403, 2019.

[299] C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov, and M. Naor, "Our data, ourselves: Privacy via distributed noise generation," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, Springer, pp. 486–503, 2006.

[300] C. Sabater, A. Bellet, and J. Ramon, "Distributed differentially private averaging with improved utility and robustness to malicious parties," *arXiv preprint arXiv:2006.07218*, 2020.

[301] A. Ghosh, T. Roughgarden, and M. Sundararajan, "Universally utility-maximizing privacy mechanisms," in *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing,* STOC '09, pp. 351–360, New York, NY, USA: ACM, 2009. [Online]. Available: http://doi.acm.org/10.1145/1536414.1536464.

[302] V. Rastogi and S. Nath, "Differentially private aggregation of distributed time-series with transformation and encryption," in *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data,* SIGMOD '10, pp. 735–746, New York, NY, USA: ACM, 2010. [Online]. Available: http://doi.acm.org/10.1145/1807167.1807247.

[303] Ú. Erlingsson, V. Feldman, I. Mironov, A. Raghunathan, K. Talwar, and A. Thakurta, "Amplification by shuffling: From local to central differential privacy via anonymity," in *SODA*, pp. 2468–2479, 2019.

[304] B. Balle, J. Bell, A. Gascón, and K. Nissim, "The privacy blanket of the shuffle model," in *Advances in Cryptology – CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18–22, 2019, Proceedings, Part II*, pp. 638–667, 2019. DOI: 10.1007/978-3-030-26951-7_22.

[305] L. Chen, B. Ghazi, R. Kumar, and P. Manurangsi, "On distributed differential privacy and counting distinct elements," in *Innovations in Theoretical Computer Science (ITCS)*, 2021.

[306] B. Ghazi, R. Kumar, P. Manurangsi, and R. Pagh, "Private counting from anonymous messages: Near-optimal accuracy with vanishing communication overhead," in *ICML*, 2020.

[307] B. Ghazi, R. Pagh, and A. Velingker, "Scalable and differentially private distributed aggregation in the shuffled model," *arXiv preprint arXiv:1906.08320*, 2019.

[308] B. Ghazi, P. Manurangsi, R. Pagh, and A. Velingker, "Private aggregation from fewer anonymous messages," in *EUROCRYPT*, pp. 798–827, 2020.

[309] B. Ghazi, N. Golowich, R. Kumar, R. Pagh, and A. Velingker, "On the power of multiple anonymous messages," *arXiv:1908.11358*, 2019.

[310] B. Ghazi, N. Golowich, R. Kumar, P. Manurangsi, R. Pagh, and A. Velingker, "Pure differentially private summation from anonymous messages," in *ITC*, pp. 15:1–15:23, 2020.

[311] B. Avent, A. Korolova, D. Zeber, T. Hovden, and B. Livshits, "BLENDER: Enabling local search with a hybrid differential privacy model," in *26th USENIX Security Symposium (USENIX Security 17)*, pp. 747–764, Vancouver, BC: USENIX Association, Aug. 2017. [Online]. Available: https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/avent.

[312] A. Beimel, A. Korolova, K. Nissim, O. Sheffet, and U. Stemmer, "The power of synergy in differential privacy: Combining a small curator with local randomizers," in *Conference on Information-Theoretic Cryptography (ITC)*, 2020. [Online]. Available: https://arxiv.org/abs/1912.08951.

[313] L. Babai, L. Fortnow, L. A. Levin, and M. Szegedy, "Checking computations in polylogarithmic time," in *STOC*, pp. 21–31, ACM, 1991.

[314] S. Micali, "Computationally sound proofs," *SIAM J. Comput.*, vol. 30, no. 4, pp. 1253–1298, 2000.

[315] S. Goldwasser, Y. T. Kalai, and G. N. Rothblum, "Delegating computation: Interactive proofs for muggles," in *STOC*, pp. 113–122, ACM, 2008.

[316] R. Gennaro, C. Gentry, and B. Parno, "Non-interactive verifiable computing: Outsourcing computation to untrusted workers," in *CRYPTO,* Lecture Notes in Computer Science, vol. 6223, pp. 465–482, Springer, 2010.

[317] S. Goldwasser, S. Micali, and C. Rackoff, "The knowledge complexity of interactive proof systems," *SIAM J. Comput.*, vol. 18, no. 1, pp. 186–208, 1989.

[318] B. Parno, J. Howell, C. Gentry, and M. Raykova, "Pinocchio: Nearly practical verifiable computation," *Commun. ACM*, vol. 59, no. 2, pp. 103–112, 2016.

[319] C. P. Schnorr, "Efficient identification and signatures for smart cards," in *Proceedings of the Workshop on the Theory and Application of Cryptographic Techniques on Advances in Cryptology,* EUROCRYPT '89, 1990.

[320] Damgård, *On σ protocols*, http://www.cs.au.dk/~ivan/Sigma.pdf, 2010.

[321] N. Bitansky, R. Canetti, A. Chiesa, and E. Tromer, "From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again," in *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference,* ITCS '12, 2012.

[322] R. Gennaro, C. Gentry, B. Parno, and M. Raykova, "Quadratic span programs and succinct NIZKs without PCPs," in *EUROCRYPT,* Lecture Notes in Computer Science, vol. 7881, pp. 626–645, Springer, 2013.

[323] C. Costello, C. Fournet, J. Howell, M. Kohlweiss, B. Kreuter, M. Naehrig, B. Parno, and S. Zahur, "Geppetto: Versatile verifiable computation," in *IEEE Symposium on Security and Privacy*, pp. 253–270, IEEE Computer Society, 2015.

[324] *libsnark: A c++ library for zkSNARK proofs*, https://github.com/scipr-lab/libsnark, Dec. 2019.

[325] E. Ben-Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza, "Zerocash: Decentralized anonymous payments from bitcoin," in *IEEE Symposium on Security and Privacy*, pp. 459–474, IEEE Computer Society, 2014.

[326] T. Xie, J. Zhang, Y. Zhang, C. Papamanthou, and D. Song, "Libra: Succinct zero-knowledge proofs with optimal prover computation," in *CRYPTO (3),* Lecture Notes in Computer Science, vol. 11694, pp. 733–764, Springer, 2019.

[327] S. Ames, C. Hazay, Y. Ishai, and M. Venkitasubramaniam, "Ligero: Lightweight sublinear arguments without a trusted setup," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security,* CCS '17, 2017.

[328] B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell, "Bulletproofs: Short proofs for confidential transactions and more," in *2018 IEEE Symposium on Security and Privacy, SP 2018, Proceedings, 21–23 May 2018, San Francisco, California, USA*, 2018.

[329] R. S. Wahby, I. Tzialla, A. Shelat, J. Thaler, and M. Walfish, "Doubly-efficient zksnarks without trusted setup," in *2018 IEEE Symposium on Security and Privacy, SP 2018, Proceedings, 21–23 May 2018, San Francisco, California, USA*, 2018.

[330] E. Ben-Sasson, I. Bentov, Y. Horesh, and M. Riabzev, "Scalable zero knowledge with no trusted setup," in *CRYPTO (3),* Lecture Notes in Computer Science, vol. 11694, pp. 701–732, Springer, 2019.

[331] D. Boneh, E. Boyle, H. Corrigan-Gibbs, N. Gilboa, and Y. Ishai, "Zero-knowledge proofs on secret-shared data via fully linear PCPs," in *CRYPTO (3),* Lecture Notes in Computer Science, vol. 11694, pp. 67–97, Springer, 2019.

[332] F. Tramèr, F. Zhang, H. Lin, J. Hubaux, A. Juels, and E. Shi, "Sealed-glass proofs: Using transparent enclaves to prove and sell knowledge," in *2017 IEEE European Symposium on Security and Privacy, EuroS&P 2017, Paris, France, April 26–28, 2017*, pp. 19–34, 2017.

[333] A. Seshadri, M. Luk, A. Perrig, L. van Doom, and P. K. Khosla, "Pioneer: Verifying code integrity and enforcing untampered code execution on legacy systems," in *Malware Detection,* Advances in Information Security, vol. 27, Springer, 2007, pp. 253–289.

[334] K. Eldefrawy, G. Tsudik, A. Francillon, and D. Perito, "SMART: Secure and minimal architecture for (establishing dynamic) root of trust," in *NDSS*, The Internet Society, 2012.

[335] P. Koeberl, S. Schulz, A.-R. Sadeghi, and V. Varadharajan, "TrustLite: A security architecture for tiny embedded devices," in *EuroSys*, 10:1–10:14, ACM, 2014.

[336] A. Francillon, Q. Nguyen, K. B. Rasmussen, and G. Tsudik, "A minimalist approach to remote attestation," in *DATE*, pp. 1–6, European Design and Automation Association, 2014.

[337] N. Carlini, C. Liu, J. Kos, Ú. Erlingsson, and D. Song, "The secret sharer: Measuring unintended neural network memorization and extracting secrets," *arXiv preprint arXiv:1802.08232*, 2018.

[338] N. Carlini, F. Tramèr, E. Wallace, M. Jagielski, A. Herbert-Voss, K. Lee, A. Roberts, T. Brown, D. Song, Ú. Erlingsson, A. Oprea, and C. Raffel, "Extracting training data from large language models," *arXiv preprint arXiv:2012.07805*, 2020.

[339] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, ACM, pp. 1322–1333, 2015.

[340] L. Melis, C. Song, E. De Cristofaro, and V. Shmatikov, “Exploiting unintended feature leakage in collaborative learning,” *arXiv preprint arXiv:1805.04049*, 2018.

[341] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, “Membership inference attacks against machine learning models,” in *2017 IEEE Symposium on Security and Privacy (SP)*, IEEE, pp. 3–18, 2017.

[342] S. Yeom, I. Giacomelli, M. Fredrikson, and S. Jha, “Privacy risk in machine learning: Analyzing the connection to overfitting,” in *2018 IEEE 31st Computer Security Foundations Symposium (CSF)*, IEEE, pp. 268–282, 2018.

[343] M. Diaz, P. Kairouz, J. Liao, and L. Sankar, “Theoretical guarantees for model auditing with finite adversaries,” *arXiv preprint arXiv:1911.03405*, 2019.

[344] A. Bhowmick, J. Duchi, J. Freudiger, G. Kapoor, and R. Rogers, “Protection against reconstruction and its applications in private federated learning,” *arXiv preprint arXiv:1812.00984*, 2018.

[345] H. B. McMahan, G. Andrew, Ú. Erlingsson, S. Chien, I. Mironov, N. Papernot, and P. Kairouz, “A general approach to adding differential privacy to iterative training procedures,” *arXiv preprint arXiv:1812.06210*, 2018.

[346] C. Dwork, G. N. Rothblum, and S. Vadhan, “Boosting and differential privacy,” in *Proceedings of the IEEE 51st Annual Symposium on Foundations of Computer Science,* FOCS ’10, pp. 51–60, 2010.

[347] P. Kairouz, S. Oh, and P. Viswanath, “The composition theorem for differential privacy,” *IEEE Trans. Inform. Theor.*, vol. 63, no. 6, pp. 4037–4049, 2017.

[348] I. Mironov, “Rényi differential privacy,” in *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*, IEEE, pp. 263–275, 2017.

[349] I. Mironov, K. Talwar, and L. Zhang, “Rényi differential privacy of the sampled Gaussian mechanism,” *arXiv preprint arXiv:1908.10530*, 2019.

[350] Y.-X. Wang, B. Balle, and S. Kasiviswanathan, “Subsampled Rényi differential privacy and analytical moments accountant,” *arXiv preprint arXiv:1808.00087*, 2018.

[351] S. Ramaswamy, O. Thakkar, R. Mathews, G. Andrew, H. B. McMahan, and F. Beaufays, “Training production language models without memorizing user data,” *arXiv preprint arXiv:2009.10031*, 2020.

[352] N. Papernot, A. Thakurta, S. Song, S. Chien, and Ú. Erlingsson, “Tempered sigmoid activations for deep learning with differential privacy,” *arXiv preprint arXiv:2007.14191*, 2020.

[353] F. Tramèr and D. Boneh, “Differentially private learning needs better features (or much more data),” *arXiv preprint arXiv:2011.11660*, 2020.

[354] K. Amin, A. Kulesza, A. Munoz, and S. Vassilvtiskii, “Bounding user contributions: A bias-variance trade-off in differential privacy,” in *International Conference on Machine Learning*, pp. 263–271, 2019.

[355] V. Pichapati, A. T. Suresh, F. X. Yu, S. J. Reddi, and S. Kumar, “AdaCliP: Adaptive clipping for private SGD,” *arXiv preprint arXiv:1908.07643*, 2019.

[356] Y. Liu, A. T. Suresh, F. X. X. Yu, S. Kumar, and M. Riley, “Learning discrete distributions: User vs item-level privacy,” in *Advances in Neural Information Processing Systems*, vol. 33, pp. 20 965–20 976, 2020.

[357] V. Feldman, I. Mironov, K. Talwar, and A. Thakurta, “Privacy amplification by iteration,” in *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, IEEE, pp. 521–532, 2018.

[358] B. Balle, P. Kairouz, H. B. McMahan, O. Thakkar, and A. Thakurta, “Privacy amplification via random check-ins,” in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, Eds., vol. 33, pp. 4623–4634, Curran Associates, Inc., 2020.

[359] P. Kairouz, B. McMahan, S. Song, O. Thakkar, A. Thakurta, and Z. Xu, *Practical and private (deep) learning without sampling or shuffling*, 2021.

[360] J. K. Salmon, M. A. Moraes, R. O. Dror, and D. E. Shaw, “Parallel random numbers: As easy as 1, 2, 3,” in *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, ACM, p. 16, 2011.

[361] I. Mironov, O. Pandey, O. Reingold, and S. Vadhan, “Computational differential privacy,” in *Advances in Cryptology—CRYPTO*, pp. 126–142, 2009.

[362] A. Haeberlen, B. C. Pierce, and A. Narayan, “Differential privacy under fire.,” in *USENIX Security Symposium*, 2011.

[363] I. Mironov, “On significance of the least significant bits for differential privacy,” in *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, ACM, pp. 650–661, 2012.

[364] Z. Ding, Y. Wang, G. Wang, D. Zhang, and D. Kifer, “Detecting violations of differential privacy,” in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, CCS ’18, pp. 475–489, New York, NY, USA: ACM, 2018. [Online]. Available: http://doi.acm.org/10.1145/3243734.3243818.

[365] M. Jagielski, J. Ullman, and A. Oprea, “Auditing differentially private machine learning: How private is private SGD?” in *Advances in Neural Information Processing Systems*, vol. 33, pp. 22 205–22 216, 2020.

[366] X. Liu and S. Oh, “Minimax rates of estimating approximate differential privacy,” *arXiv preprint arXiv:1905.10335*, 2019.

[367] M. S. Riazi, K. Laine, B. Pelton, and W. Dai, “HEAX: High-performance architecture for computation on homomorphically encrypted data in the cloud,” *arXiv preprint arXiv:1909.09731*, 2019.

[368] R. Cummings, S. Krehbiel, Y. Mei, R. Tuo, and W. Zhang, “Differentially private change-point detection,” in *Advances in Neural Information Processing Systems,* NeurIPS ’18, vol. 31, pp. 10 825–10 834, 2018.

[369] R. Cummings, S. Krehbiel, K. Lai, and U. Tantitongpipat, “Differential privacy for growing databases,” in *Advances in Neural Information Processing Systems,* NeurIPS ’18, vol. 31, pp. 8864–8873, 2018.

[370] C. L. Canonne, G. Kamath, A. McMillan, A. Smith, and J. Ullman, "The structure of optimal private tests for simple hypotheses," *arXiv preprint arXiv:1811.11148*, 2019.

[371] N. C. Abay, Y. Zhou, M. Kantarcioglu, B. Thuraisingham, and L. Sweeney, "Privacy preserving synthetic data release using deep learning," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, Springer, pp. 510–526, 2018.

[372] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing machine learning models via prediction APIs," in *25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10–12, 2016.*, pp. 601–618, 2016. [Online]. Available: https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/tramer.

[373] J. R. Douceur, "The sybil attack," in *Revised Papers from the First International Workshop on Peer-to-Peer Systems,* IPTPS '01, pp. 251–260, London, UK: Springer-Verlag, 2002. [Online]. Available: http://dl.acm.org/citation.cfm?id=646334.687813.

[374] R. Bassily, N. Kobbi, U. Stemmer, and A. G. Thakurta, "Practical locally private heavy hitters," *J. Mach. Learn. Res.*, vol. 21, no. 16, pp. 1–42, 2020.

[375] G. Cormode, T. Kulkarni, and D. Srivastava, "Marginal release under local differential privacy," in *Proceedings of the 2018 International Conference on Management of Data*, ACM, pp. 131–146, 2018.

[376] J. C. Duchi, M. I. Jordan, and M. J. Wainwright, "Local privacy and statistical minimax rates," in *Foundations of Computer Science (FOCS), 2013 IEEE 54th Annual Symposium on*, IEEE, pp. 429–438, 2013.

[377] P. Kairouz, S. Oh, and P. Viswanath, "Extremal mechanisms for local differential privacy," in *Advances in Neural Information Processing Systems*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds., vol. 27, pp. 2879–2887, Curran Associates, Inc., 2014.

[378] P. Kairouz, K. A. Bonawitz, and D. Ramage, "Discrete distribution estimation under local privacy," in *International Conference on Machine Learning*, pp. 2436–2444, 2016.

[379] M. Ye and A. Barg, "Optimal schemes for discrete distribution estimation under locally differential privacy," *IEEE Trans. Inform. Theor.*, 2018.

[380] J. Tang, A. Korolova, X. Bai, X. Wang, and X. Wang, "Privacy loss in Apple's implementation of differential privacy on MacOS 10.12," *CoRR*, vol. abs/1709.02753, 2017. [Online]. Available: http://arxiv.org/abs/1709.02753.

[381] B. Avent, Y. Dubey, and A. Korolova, "The power of the hybrid model for mean estimation," *Proceedings on Privacy Enhancing Technologies (PETS)*, vol. 2020, no. 4, pp. 48–68, 1 Oct. 2020. DOI: https://doi.org/10.2478/popets-2020-0062.

[382] E. Kushilevitz and N. Nisan, *Communication Complexity*. New York, NY, USA: Cambridge University Press, 1997.

[383] W.-N. Chen, P. Kairouz, and A. Ozgur, "Breaking the communication-privacy-accuracy trilemma," in *Advances in Neural Information Processing Systems*, vol. 33, pp. 3312–3324, 2020.

[384] B. Balle, J. Bell, A. Gascón, and K. Nissim, "Private summation in the multi-message shuffle model," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, ACM, pp. 657–676, 2020.

[385] R. Bassily and A. Smith, "Local, private, efficient protocols for succinct histograms," in *STOC*, pp. 127–135, 2015.

[386] F. McSherry and K. Talwar, "Mechanism design via differential privacy.," in *FOCS*, pp. 94–103, 2007.

[387] T. Steinke and J. Ullman, "Tight lower bounds for differentially private selection," in *FOCS*, pp. 552–563, 2017.

[388] A. M. Girgis, D. Data, S. Diggavi, P. Kairouz, and A. T. Suresh, "Shuffled model of federated learning: Privacy, communication and accuracy trade-offs," *arXiv preprint arXiv:2008.07180*, 2020.

[389] P. Kairouz, Z. Liu, and T. Steinke, "The distributed discrete gaussian mechanism for federated learning with secure aggregation," *arXiv preprint arXiv:2102.06387*, 2021.

[390] W.-T. Chang and R. Tandon, "On the upload versus download cost for secure and private matrix multiplication," *ArXiv, abs/1906.10684*, 2019.

[391] Z. Jia and S. A. Jafar, "On the capacity of secure distributed matrix multiplication," *CoRR*, vol. abs/1908.06957, 2019.

[392] C. Niu, F. Wu, S. Tang, L. Hua, R. Jia, C. Lv, Z. Wu, and G. Chen, "Secure federated submodel learning," *arXiv preprint arXiv:1911.02254*, 2019.

[393] M. J. Kearns, A. Roth, Z. S. Wu, and G. Yaroslavtsev, "Privacy for the protected (only)," *CoRR*, vol. abs/1506.00242, 2015. [Online]. Available: http://arxiv.org/abs/1506.00242.

[394] D. Kifer and A. Machanavajjhala, "Pufferfish: A framework for mathematical privacy definitions," *ACM Trans. Database Sys.*, vol. 39, no. 1, 3:1–3:36, 2014.

[395] J. M. Abowd and I. M. Schmutte, "An economic analysis of privacy protection and statistical accuracy as social choices," *Am. Econ. Rev.*, vol. 109, no. 1, pp. 171–202, 2019.

[396] R. Cummings, I. Dekel, O. Heffetz, and K. Ligett, "Bringing differential privacy into the experimental economics lab: Theory and an application to a public-good game," Working paper, 2019.

[397] B. Biggio, B. Nelson, and P. Laskov, "Poisoning attacks against support vector machines," in *Proceedings of the 29th International Coference on International Conference on Machine Learning,* ICML'12, pp. 1467–1474, Edinburgh, UK: Omnipress, 2012. [Online]. Available: http://dl.acm.org/citation.cfm?id=3042573.3042761.

[398] Y. Liu, S. Ma, Y. Aafer, W.-C. Lee, J. Zhai, W. Wang, and X. Zhang, "Trojaning attack on neural networks," in *25th Annual Network and Distributed System Security Symposium, NDSS 2018, San Diego, California, USA, February 18–21, 2018*, 2018.

[399] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," *arXiv preprint arXiv:1807.00459*, 2018.

[400] A. N. Bhagoji, S. Chakraborty, P. Mittal, and S. Calo, "Analyzing federated learning through an adversarial lens," in *Proceedings of the 36th International Conference on Machine Learning*, pp. 634–643, 2019.

[401] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7–9, 2015, Conference Track Proceedings*, 2015. [Online]. Available: http://arxiv.org/abs/1412.6572.

[402] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *ICLR*, 2013.

[403] X. Chen, C. Liu, B. Li, K. Lu, and D. Song, "Targeted backdoor attacks on deep learning systems using data poisoning," *arXiv preprint arXiv:1712.05526*, 2017.

[404] D. Alistarh, Z. Allen-Zhu, and J. Li, "Byzantine stochastic gradient descent," in *NIPS*, 2018.

[405] P. Blanchard, E. M. El Mahdi, R. Guerraoui, and J. Stainer, "Machine learning with adversaries: Byzantine tolerant gradient descent," in *Advances in Neural Information Processing Systems*, pp. 118–128, 2017.

[406] Y. Chen, L. Su, and J. Xu, "Distributed statistical machine learning in adversarial settings: Byzantine gradient descent," *POMACS*, vol. 1, pp. 44:1–44:25, 2017.

[407] L. Chen, H. Wang, Z. B. Charles, and D. S. Papailiopoulos, "DRACO: Byzantine-resilient distributed training via redundant gradients," in *Proceedings of the 35th International Conference on Machine Learning, ICML*, 2018.

[408] E. M. El Mhamdi, R. Guerraoui, and S. Rouault, "The hidden vulnerability of distributed learning in Byzantium," in *ICML*, 2018.

[409] G. F. Cretu, A. Stavrou, M. E. Locasto, S. J. Stolfo, and A. D. Keromytis, "Casting out demons: Sanitizing training data for anomaly sensors," in *2008 IEEE Symposium on Security and Privacy (sp 2008)*, IEEE, pp. 81–95, 2008.

[410] J. Steinhardt, P. W. W. Koh, and P. S. Liang, "Certified defenses for data poisoning attacks," in *Advances in Neural Information Processing Systems*, pp. 3517–3529, 2017.

[411] I. Diakonikolas, G. Kamath, D. Kane, J. Li, J. Steinhardt, and A. Stewart, "Sever: A robust meta-algorithm for stochastic optimization," in *Proceedings of the 36th International Conference on Machine Learning,* Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97, pp. 1596–1606, Long Beach, California, USA: PMLR 9–15 Jun. 2019, Sep. 2019. [Online]. Available: http://proceedings.mlr.press/v97/diakonikolas19a.html.

[412] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Šrndić, P. Laskov, G. Giacinto, and F. Roli, "Evasion attacks against machine learning at test time," in *ECML-PKDD*, Springer, pp. 387–402, 2013.

[413] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *2017 IEEE Symposium on Security and Privacy (SP)*, IEEE, pp. 39–57, 2017.

[414] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *ICLR*, 2015.

[415] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," *ICLR*, 2017.

[416] L. Lamport, R. Shostak, and M. Pease, "The Byzantine generals problem," *ACM Trans. Program. Lang. Syst.*, vol. 4, no. 3, pp. 382–401, 1982.

[417] D. Yin, Y. Chen, K. Ramchandran, and P. Bartlett, "Byzantine-robust distributed learning: Towards optimal statistical rates," in *ICML*, 2019.

[418] L. Su and N. H. Vaidya, "Fault-Tolerant Multi-Agent Optimization: Optimal Iterative Distributed Algorithms," in *PODC*, 2016.

[419] J. So, B. Guler, and A. S. Avestimehr, "Byzantine-resilient secure federated learning," in *IEEE Journal on Selected Areas in Communication, Series on Machine Learning for Communications and Networks*, 2020.

[420] K. Pillutla, S. M. Kakade, and Z. Harchaoui, “Robust aggregation for federated learning,” *arXiv preprint arXiv:1912.13445*, 2019.

[421] C. Xie, S. Koyejo, and I. Gupta, “Practical distributed learning: Secure machine learning with communication-efficient local updates,” in *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD)*, 2019.

[422] M. Fang, X. Cao, J. Jia, and N. Z. Gong, “Local model poisoning attacks to Byzantine-robust federated learning,” *arXiv preprint arXiv:1911.11815*, 2019.

[423] M. Baruch, G. Baruch, and Y. Goldberg, “A little is enough: Circumventing defenses for distributed learning,” *arXiv preprint arXiv:1902.06156*, 2019.

[424] S. Rajput, H. Wang, Z. Charles, and D. Papailiopoulos, “DETOX: A redundancy-based framework for faster and more robust gradient aggregation,” *arXiv preprint arXiv:1907.12205*, 2019.

[425] D. Data, L. Song, and S. Diggavi, “Data encoding for byzantine-resilient distributed optimization,” *IEEE Trans. Inform. Theor.*, pp. 2719–2723, 2020.

[426] C.-L. Chen, L. Golubchik, and M. Paolieri, “Backdoor attacks on federated meta-learning,” *arXiv preprint arXiv:2006.07026*, 2020.

[427] E. Chou, F. Tramèr, and G. Pellegrino, “SentiNet: Detecting physical attacks against deep learning systems,” *arXiv preprint arXiv:1812.00292*, 2018.

[428] K. Liu, B. Dolan-Gavitt, and S. Garg, “Fine-pruning: Defending against backdooring attacks on deep neural networks,” in *International Symposium on Research in Attacks, Intrusions, and Defenses*, Springer, pp. 273–294, 2018.

[429] Y. Shen and S. Sanghavi, “Learning with bad training data via iterative trimmed loss minimization,” in *Proceedings of the 36th International Conference on Machine Learning,* Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97, pp. 5739–5748, Long Beach, California, USA: PMLR. 9–15 Jun. 2019, 2019. [Online]. Available: http://proceedings.mlr.press/v97/shen19e.html.

[430] B. Tran, J. Li, and A. Madry, “Spectral signatures in backdoor attacks,” in *Advances in Neural Information Processing Systems*, pp. 8000–8010, 2018.

[431] B. Wang, Y. Yao, S. Shan, H. Li, B. Viswanath, H. Zheng, and B. Y. Zhao, “Neural cleanse: Identifying and mitigating backdoor attacks in neural networks,” in *2019 IEEE Symposium on Security and Privacy*, IEEE, 2019.

[432] C. Fung, C. J.-M. Yoon, and I. Beschastnikh, “Mitigating sybils in federated learning poisoning,” *arXiv preprint arXiv:1808.04866*, 2018.

[433] P. W. Koh and P. Liang, “Understanding black-box predictions via influence functions,” in *Proceedings of the 34th International Conference on Machine Learning – Volume 70*, JMLR. org, pp. 1885–1894, 2017.

[434] C. Xie, S. Koyejo, and I. Gupta, “Zeno: Distributed stochastic gradient descent with suspicion-based fault-tolerance,” in *International Conference on Machine Learning*, pp. 6893–6901, 2019.

[435] C. Xie, “Zeno++: Robust asynchronous SGD with arbitrary number of Byzantine workers,” *arXiv preprint arXiv:1903.07020*, 2019.

[436] T. Gu, B. Dolan-Gavitt, and S. Garg, “BadNets: Identifying vulnerabilities in the machine learning model supply chain,” *arXiv preprint arXiv:1708.06733*, 2017.

[437] P. W. Koh, J. Steinhardt, and P. Liang, “Stronger data poisoning attacks break data sanitization defenses,” *arXiv preprint arXiv:1811.00741*, 2018.

[438] X. Zhu, “Machine teaching: An inverse problem to machine learning and an approach toward optimal education,” in *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[439] S. Mei and X. Zhu, “Using machine teaching to identify optimal training-set attacks on machine learners,” in *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[440] L. Engstrom, B. Tran, D. Tsipras, L. Schmidt, and A. Madry, “A rotation and a translation suffice: Fooling CNNs with simple transformations,” *arXiv preprint arXiv:1712.02779*, 2017.

[441] D. Kang, Y. Sun, D. Hendrycks, T. Brown, and J. Steinhardt, "Testing robustness against unforeseen adversaries," *arXiv preprint arXiv:1908.08016*, 2019.

[442] E. Wong, F. R. Schmidt, and J. Z. Kolter, "Wasserstein adversarial examples via projected sinkhorn iterations," *ICML*, 2019.

[443] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial machine learning at scale," *arXiv preprint arXiv:1611.01236*, 2016.

[444] P.-Y. Chen, H. Zhang, Y. Sharma, J. Yi, and C.-J. Hsieh, "ZOO: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models," in *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, ACM, pp. 15–26, 2017.

[445] W. Brendel, J. Rauber, and M. Bethge, "Decision-based adversarial attacks: Reliable attacks against black-box machine learning models," *arXiv preprint arXiv:1712.04248*, 2017.

[446] F. Tramèr, A. Kurakin, N. Papernot, I. J. Goodfellow, D. Boneh, and P. D. McDaniel, "Ensemble adversarial training: Attacks and defenses," in *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 – May 3, 2018, Conference Track Proceedings*, 2018.

[447] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, ACM, pp. 506–519, 2017.

[448] A. Athalye, N. Carlini, and D. Wagner, "Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples," *ICML*, 2018.

[449] A. Shafahi, M. Najibi, A. Ghiasi, Z. Xu, J. Dickerson, C. Studer, L. S. Davis, G. Taylor, and T. Goldstein, "Adversarial training for free," *NeurIPS*, 2019.

[450] C. Xie, Y. Wu, L. van der Maaten, A. Yuille, and K. He, "Feature denoising for improving adversarial robustness," *CVPR*, 2019.

[451] Y. Sharma and P.-Y. Chen, "Attacking the Madry defense model with L_1-based adversarial examples," *arXiv preprint arXiv:1710.10733*, 2017.

[452] F. Tramèr and D. Boneh, "Adversarial training and robustness for multiple perturbations," *arXiv preprint arXiv:1904.13000*, 2019.

[453] F. Tramèr, J. Behrmann, N. Carlini, N. Papernot, and J.-H. Jacobsen, "Fundamental tradeoffs between invariance and sensitivity to adversarial perturbations," in *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13–18 July 2020, Virtual Event,* Proceedings of Machine Learning Research, vol. 119, pp. 9561–9571, PMLR, 2020. [Online]. Available: http://proceedings.mlr.press/v119/tramer20a.html.

[454] B. Chen, W. Carvalho, N. Baracaldo, H. Ludwig, B. Edwards, T. Lee, I. Molloy, and B. Srivastava, "Detecting backdoor attacks on deep neural networks by activation clustering," *arXiv preprint arXiv:1811.03728*, 2018.

[455] Z. Sun, P. Kairouz, A. T. Suresh, and H. B. McMahan, "Can you really backdoor federated learning?" *arXiv preprint arXiv:1911.07963*, 2019.

[456] H. Wang, K. Sreenivasan, S. Rajput, H. Vishwakarma, S. Agarwal, J.-y. Sohn, K. Lee, and D. Papailiopoulos, "Attack of the tails: Yes, you really can backdoor federated learning," *arXiv preprint arXiv:2007.05084*, 2020.

[457] Y. Ma, X. Zhu, and J. Hsu, "Data poisoning against differentially-private learners: Attacks and defenses," in *International Joint Conference on Artificial Intelligence (IJCAI), Macao, China*, 2019. [Online]. Available: https://arxiv.org/abs/1903.09860.

[458] R. C. Geyer, T. Klein, and M. Nabi, "Differentially private federated learning: A client level perspective," *CoRR*, vol. abs/1712.07557, 2017. [Online]. Available: http://arxiv.org/abs/1712.07557.

[459] M. Lécuyer, V. Atlidakis, R. Geambasu, D. Hsu, and S. Jana, "Certified robustness to adversarial examples with differential privacy," in *2019 IEEE Symposium on Security and Privacy, SP 2019, San Francisco, CA, USA, May 19–23, 2019*, pp. 656–672, 2019. DOI: 10.1109/SP.2019.00044.

[460] K. Srinathan and C. P. Rangan, "Efficient asynchronous secure multiparty distributed computation," in *International Conference on Cryptology in India*, Springer, pp. 117–129, 2000.

[461] V. Mnih and G. E. Hinton, "Learning to label aerial images from noisy data," in *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, pp. 567–574, 2012.

[462] N. Natarajan, I. S. Dhillon, P. K. Ravikumar, and A. Tewari, "Learning with noisy labels," in *Advances in Neural Information Processing Systems*, pp. 1196–1204, 2013.

[463] C. Dwork, M. Hardt, T. Pitassi, O. Reingold, and R. Zemel, "Fairness through awareness," in *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ACM, pp. 214–226, 2012.

[464] S. Barocas, M. Hardt, and A. Narayanan, *Fairness and Machine Learning*. fairmlbook.org, 2019.

[465] S. Mitchell, E. Potash, and S. Barocas, "Prediction-based decisions and fairness: A catalogue of choices, assumptions, and definitions," *arXiv preprint arXiv:1811.07867*, 2018.

[466] M. J. Kusner, J. Loftus, C. Russell, and R. Silva, "Counterfactual fairness," in *Advances in Neural Information Processing Systems*, pp. 4066–4076, 2017.

[467] T. Kamishima, S. Akaho, and J. Sakuma, "Fairness-aware learning through regularization approach," in *2011 IEEE 11th International Conference on Data Mining Workshops*, IEEE, pp. 643–650, 2011.

[468] J. Buolamwini and T. Gebru, "Gender shades: Intersectional accuracy disparities in commercial gender classification," in *Conference on Fairness, Accountability and Transparency*, pp. 77–91, 2018.

[469] T. Li, M. Sanjabi, and V. Smith, "Fair resource allocation in federated learning," *arXiv preprint arXiv:1905.10497*, 2019.

[470] L. Eckhouse, K. Lum, C. Conti-Cook, and J. Ciccolini, "Layers of bias: A unified approach for understanding problems with risk assessment," *Criminal Justice and Behavior*, vol. 46, no. 2, pp. 185–209, 2019.

[471] R. Richardson, J. Schultz, and K. Crawford, "Dirty data, bad predictions: How civil rights violations impact police data, predictive policing systems, and justice," *New York University Law Review Online, Forthcoming*, 2019.

[472] N. Sambasivan, G. Checkley, A. Batool, N. Ahmed, D. Nemer, L. S. Gaytán-Lugo, T. Matthews, S. Consolvo, and E. Churchill, "'privacy is not for me, it's for those rich women': Performative privacy practices on mobile phones by women in South Asia," in *Fourteenth Symposium on Usable Privacy and Security (SOUPS 2018)*, pp. 127–142, 2018.

[473] P. Kairouz, J. Liao, C. Huang, and L. Sankar, "Censored and fair universal representations using generative adversarial models," *arXiv preprint arXiv:1910.00411*, 2020.

[474] T. Hashimoto, M. Srivastava, H. Namkoong, and P. Liang, "Fairness without demographics in repeated loss minimization," in *International Conference on Machine Learning*, pp. 1934–1943, 2018.

[475] Ú. Hébert-Johnson, M. Kim, O. Reingold, and G. Rothblum, "Multicalibration: Calibration for the (computationally-identifiable) masses," in *International Conference on Machine Learning*, pp. 1944–1953, 2018.

[476] R. Cummings, V. Gupta, D. Kimpara, and J. Morgenstern, "On the compatibility of privacy and fairness," in *Proceedings of Fairness in User Modeling, Adaptation and Personalization,* FairUMAP, 2019.

[477] M. Jagielski, M. J. Kearns, J. Mao, A. Oprea, A. Roth, S. Sharifi-Malvajerdi, and J. Ullman, "Differentially private fair learning," *CoRR*, vol. abs/1812.02696, 2018. [Online]. Available: http://arxiv.org/abs/1812.02696.

[478] E. Bagdasaryan and V. Shmatikov, "Differential privacy has disparate impact on model accuracy," *CoRR*, vol. abs/1905.12101, 2019. arXiv: 1905.12101. [Online]. Available: http://arxiv.org/abs/1905.12101.

[479] S. Kuppam, R. McKenna, D. Pujol, M. Hay, A. Machanavajjhala, and G. Miklau, "Fair decision making using privacy-protected data," *CoRR*, vol. abs/1905.12744, 2019. [Online]. Available: http://arxiv.org/abs/1905.12744.

[480] L. Song, R. Shokri, and P. Mittal, "Privacy risks of securing machine learning models against adversarial examples," in *Proceedings of the ACM Conference on Computer and Communication Security (CCS)*, 2019.

[481] M. Bertrán, N. Martínez, A. Papadaki, Q. Qiu, M. R. D. Rodrigues, G. Reeves, and G. Sapiro, "Learning adversarially fair and transferable representations," in *ICML*, 2019.

[482] C. Feutry, P. Piantanida, Y. Bengio, and P. Duhamel, "Learning anonymized representations with adversarial neural networks," *CoRR*, vol. abs/1802.09386, 2018. [Online]. Available: http://arxiv.org/abs/1802.09386.

[483] D. Madras, E. Creager, T. Pitassi, and R. Zemel, "Learning adversarially fair and transferable representations," in *ICML*, 2018.

[484] B. M. L. Srivastava, A. Bellet, M. Tommasi, and E. Vincent, "Privacy-preserving adversarial representation learning in ASR: Reality or illusion?" In *Annual Conference of the International Speech Communication Association (Interspeech)*, 2019.

[485] T. S. Brisimi, R. Chen, T. Mela, A. Olshevsky, I. C. Paschalidis, and W. Shi, "Federated learning of predictive models from federated electronic health records," *Int. J. Med. Informat.*, vol. 112, pp. 59–67, 2018.

[486] 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," *JAMIA*, vol. 25, no. 8, pp. 945–954, 2018.

[487] A. R. Martin, M. Kanai, Y. Kamatani, Y. Okada, B. M. Neale, and M. J. Daly, "Current clinical use of polygenic scores will risk exacerbating health disparities," *BioRxiv*, p. 441261, 2019.

[488] Y. Laguel, K. Pillutla, J. Malick, and Z. Harchaoui, "Device heterogeneity in federated learning: A superquantile approach," *arXiv preprint arXiv:2002.11223*, 2020.

[489] C. T. Dinh, N. H. Tran, and T. D. Nguyen, "Personalized Federated Learning with Moreau Envelopes," in *NeurIPS*, 2020.

[490] P. Awasthi, C. Cortes, Y. Mansour, and M. Mohri, "Beyond individual and group fairness," *CoRR*, vol. abs/2008.09490, 2020.

[491] M. Hardt, E. Price, and N. Srebro, "Equality of opportunity in supervised learning," in *Advances in Neural Information Processing Systems*, pp. 3323–3331, 2016.

[492] M. B. Zafar, I. Valera, M. G. Rodriguez, and K. P. Gummadi, "Fairness constraints: Mechanisms for fair classification," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, 2017.

[493] D. Ramage and S. Mazzocchi, *Federated analytics: Collaborative data science without data collection*, [Online]. Available: https://ai.googleblog.com/2020/05/federated-analytics-collaborative-data.html. Google AI Blog, May 2020.

[494] R. J. A. Little, "Post-stratification: A modeler's perspective," *J. Am. Stat. Assoc.*, vol. 88, no. 423, pp. 1001–1012, 1993.

[495] D. J. Beutel, T. Topal, A. Mathur, X. Qiu, T. Parcollet, and N. D. Lane, "Flower: A friendly federated learning research framework," *arXiv preprint arXiv:2007.14390*, 2020.

[496] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32, pp. 8024–8035, Curran Associates, Inc., 2019. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[497] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang, *JAX: Composable transformations of Python+NumPy programs*, 2018. [Online]. Available: http://github.com/google/jax.

[498] F. Seide and A. Agarwal, "CNTK: Microsoft's open-source deep-learning toolkit," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, p. 2135, New York, NY, USA: Association for Computing Machinery, 2016. DOI: 10.1145/2939672.2945397.

[499] W. Zhu, P. Kairouz, H. Sun, B. McMahan, and W. Li, "Federated heavy hitters discovery with differential privacy," *arXiv preprint arXiv:1902.08534*, 2019.

[500] C. He, S. Li, J. So, X. Zeng, M. Zhang, H. Wang, X. Wang, P. Vepakomma, A. Singh, H. Qiu, X. Zhu, J. Wang, L. Shen, P. Zhao, Y. Kang, Y. Liu, R. Raskar, Q. Yang, M. Annavaram, and S. Avestimehr, "FedML: A research library and benchmark for federated machine learning," *arXiv preprint arXiv:2007.13518*, 2020.

[501] N. Rodríguez-Barroso, G. Stipcich, D. Jiménez-López, J. A. Ruiz-Millán, E. Martínez-Cámara, G. González-Seco, M. V. Luzón, M. A. Veganzones, and F. Herrera, "Federated learning and differential privacy: Software tools analysis, the sherpa.ai FL framework and methodological guidelines for preserving data privacy," *Inform. Fusion*, vol. 64, pp. 270–292, 2020.

[502] PyVertical Authors, *PyVertical*, 2020. [Online]. Available: https://github.com/OpenMined/PyVertical.

[503] The PaddlePaddle Authors, *PaddlePaddle*, 2019. [Online]. Available: http://www.paddlepaddle.org/.

[504] The Fedlearner Authors, *Fedlearner*, 2020. [Online]. Available: https://github.com/bytedance/fedlearner.

[505] G. Cohen, S. Afshar, J. Tapson, and A. van Schaik, "EMNIST: An extension of MNIST to handwritten letters," *arXiv preprint arXiv:1702.05373*, 2017.

[506] S. Caldas, P. Wu, T. Li, J. Konecný, H. B. McMahan, V. Smith, and A. Talwalkar, "LEAF: A benchmark for federated settings," *arXiv preprint arXiv:1812.01097*, 2018.

[507] The Google-Landmark-v2 Authors, *Google landmark dataset v2*, 2019. [Online]. Available: https://github.com/cvdfoundation/google-landmark.

[508] J. Luo, X. Wu, Y. Luo, A. Huang, Y. Huang, Y. Liu, and Q. Yang, "Real-world image datasets for federated learning," *arXiv preprint arXiv:1910.11089*, 2019.

[509] G. J. Annas, "HIPAA regulations – a new era of medical-record privacy?" *NEJM*, vol. 348, no. 15, pp. 1486–1490, 2003.