Advances and Open Problems in Federated Learning

Peter Kairouz[7*] H. Brendan McMahan[7*] Brendan Avent[21] Aurélien Bellet[9]
Mehdi Bennis[19] Arjun Nitin Bhagoji[13] Keith Bonawitz[7] Zachary Charles[7]
Graham Cormode[23] Rachel Cummings[6] Rafael G.L. D'Oliveira[14]
Salim El Rouayheb[14] David Evans[22] Josh Gardner[24] Zachary Garrett[7]
Adrià Gascón[7] Badih Ghazi[7] Phillip B. Gibbons[2] Marco Gruteser[7,14]
Zaid Harchaoui[24] Chaoyang He[21] Lie He [4] Zhouyuan Huo [20]
Ben Hutchinson[7] Justin Hsu[25] Martin Jaggi[4] Tara Javidi[17] Gauri Joshi[2]
Mikhail Khodak[2] Jakub Konečný[7] Aleksandra Korolova[21] Farinaz Koushanfar[17]
Sanmi Koyejo[7,18] Tancrède Lepoint[7] Yang Liu[12] Prateek Mittal[13]
Mehryar Mohri[7] Richard Nock[1] Ayfer Özgür[15] Rasmus Pagh[7,10]
Mariana Raykova[7] Hang Qi[7] Daniel Ramage[7] Ramesh Raskar[11]
Dawn Song[16] Weikang Song[7] Sebastian U. Stich[4] Ziteng Sun[3]
Ananda Theertha Suresh[7] Florian Tramèr[15] Praneeth Vepakomma[11] Jianyu Wang[2]
Li Xiong[5] Zheng Xu[7] Qiang Yang[8] Felix X. Yu[7] Han Yu[12] Sen Zhao[7]

[1]Australian National University, [2]Carnegie Mellon University, [3]Cornell University,
[4]École Polytechnique Fédérale de Lausanne, [5]Emory University, [6]Georgia Institute of Technology,
[7]Google Research, [8]Hong Kong University of Science and Technology, [9]INRIA, [10]IT University of Copenhagen,
[11]Massachusetts Institute of Technology, [12]Nanyang Technological University, [13]Princeton University,
[14]Rutgers University, [15]Stanford University, [16]University of California Berkeley,
[17] University of California San Diego, [18]University of Illinois Urbana-Champaign, [19]University of Oulu,
[20]University of Pittsburgh, [21]University of Southern California, [22]University of Virginia,
[23]University of Warwick, [24]University of Washington, [25]University of Wisconsin–Madison

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 paper 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. Correspondence to kairouz@google.com and mcmahan@google.com.

Contents

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 paper 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. [289]: "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 [340, 421], and Agrawal and Srikant [15] and Vaidya et al. [390] 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 paper 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.

Since the term federated learning was initially introduced with an emphasis on mobile and edge device applications [289, 287], 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,*

[1] During the preparation of this work, Li et al. [262] independently released an excellent but less comprehensive survey.

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 earlier 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 Section 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. [156, 135]). 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 [323, 196, 420, 98, 329], as well as in features on Pixel phones [18] and in Android Messages [375]. 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 [27], for applications like the QuickType keyboard and the vocal classifier for "Hey Siri" [28]; doc.ai is developing cross-device FL solutions for medical research [130], and Snips has explored cross-device FL for hotword detection [259].

Cross-silo applications have also been proposed or described in myriad domains including finance risk prediction for reinsurance [407], pharmaceuticals discovery [158], electronic health records mining [162], medical data segmentation [19, 121], and smart manufacturing [305].

The growing demand for federated learning technology has resulted in a number of tools and frameworks becoming available. These include TensorFlow Federated [38], Federated AI Technology Enabler [34], PySyft [342], Leaf [35], PaddleFL [36] and Clara Training Framework [33]; more details in Appendix A. Commercial data platforms incorporating federated learning are in development from established technology companies as well as smaller start-ups.

Table 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 paper, 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. [74] 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 in a real-world system. We begin by sketching the lifecycle of a model before considering a FL training process.

	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).	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.
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 / repartitioned arbitrarily across clients.	Partition is fixed. Could be example-partitioned (horizontal) or feature-partitioned (vertical).	Fixed partitioning by example (horizontal).

Table 1: Typical characteristics of federated learning settings vs. distributed learning in the datacenter (e.g. [131]). 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 highlighed in bold, but the other characteristics are also critical in determining which techniques are applicable.

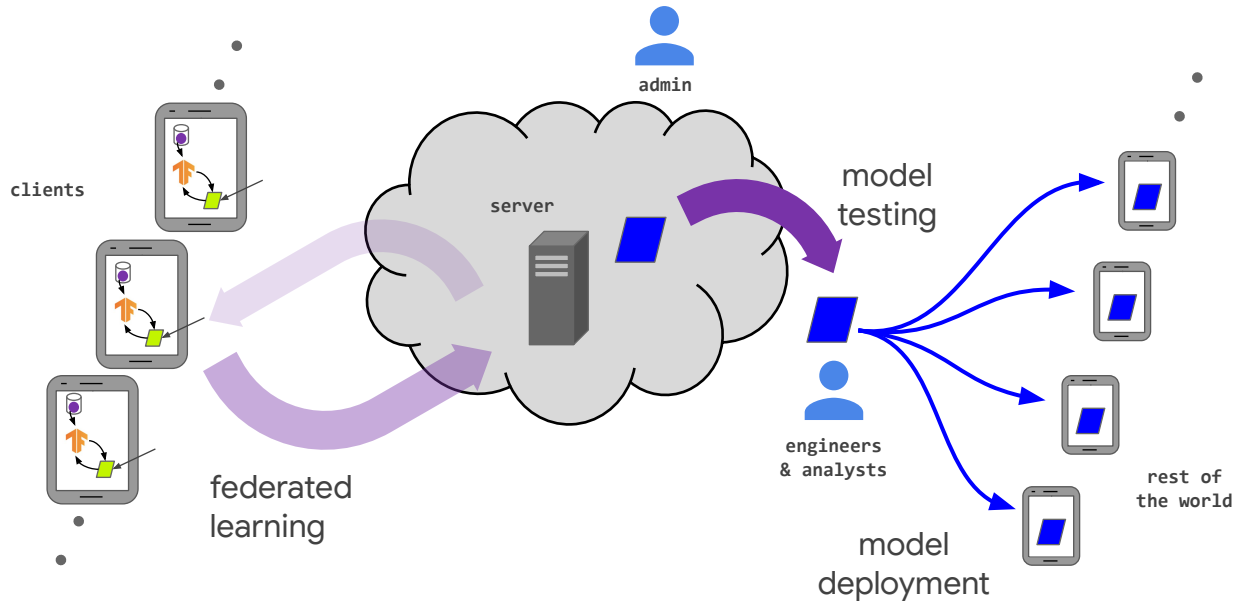


Figure 1: The lifecycle of an FL-trained model and the various actors in a federated learning system. This figure is revisited in Section 4 from a threat models perspective.

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 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

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

Table 2: Order-of-magnitude sizes for typical cross-device federated learning applications.

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 paper concerns federated training specifically, there are many other components including federated analytics tasks like model evaluation and debugging. Improving these is the focus of Section 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. [289] 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 [6]) 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 2 gives typical order-of-magnitude sizes for the quantities involved in a typical federated learning application on mobile devices.

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 paper 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, 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 paper 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 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, 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 paper, 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 [392]. Furthermore, the server may even become a bottleneck when the number of clients is very large, as demonstrated by Lian et al. [266] (though this can be mitigated by careful system design, e.g. [74]).

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[2]. 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 [30] as well as for decentralized networks of devices [110, 392, 379, 54, 243, 253, 153]. 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 [30, 200].

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 Section 2.1.2).

Table 3 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 paper as well; in this section challenges

[2]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 [78].

	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	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 3: 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 comparing cross-silo and cross-device FL.

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 [201], 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. [427] 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 [379], as we outline in more detail below.

Well-connected or denser networks encourage faster consensus and give better error convergence rates – which depend on the spectral gap of the network graph – but they 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. [401] 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 [110, 392, 54, 30, 267].

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 Section 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 [243, 242]. The convergence analysis for the case with several local update steps has recently been provided by Wang and Joshi [399]. Further, [401] 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 [392, 54] 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. [431] 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 to the fully decentralized setting, without negatively impacting the convergence is an active research direction [243, 335, 380, 242]. A complementary idea is to design decentralized optimization algorithms which naturally give rise to sparse updates [431].

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.

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 [409], 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. [73], 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 [104] 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 Section 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 [290] 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 decentralized federated learning case, we could also have each client add noise locally, as done in [54]. That is, each client locally adds a certain amount of Gaussian noise after local gradient descent steps, and submits the model to blockchain. The locally added noise scale is calculated such that the aggregated noise on blockchain is able to achieve the same client-level differential privacy as in [290]. 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, 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 — 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-data silo, 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. [419].

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 [419, Section 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 [103], linear and logistic regression [419, 198], and neural networks [276].

Federated transfer learning [419] 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 [314] to build models collaboratively. The existing formulation is limited to the case of 2 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 [407], hospitals can build better diagnostic models [121], and so on.

An open-source platform supporting the above outlined applications is currently available as *Federated AI Technology Enabler (FATE)* [34]. At the same time, the IEEE P3652.1 Federated Machine Learning Working Group is focusing on standard-setting for the Federated AI Technology Framework.

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. [225, 224]), but is particularly relevant in the cross-silo setting, where participants may also be business competitors. 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 Section 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.

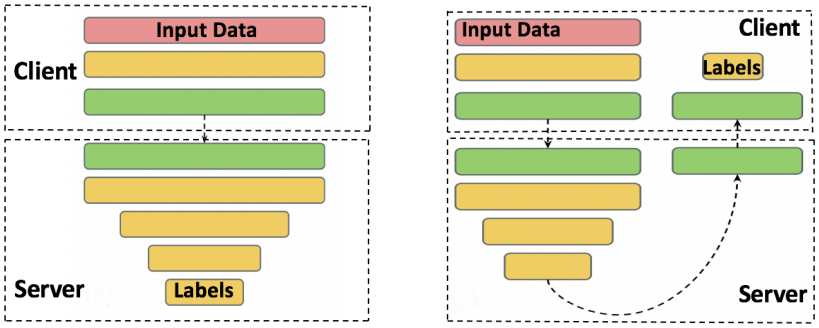
On the other hand, for a practically persuasive claim, we would usually need a notion of local differential privacy, as the potential threat from other clients is likely to be more important. In cases when the clients are not considered a significant threat, each client could control the data from a 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, [236] used an alternating direction method of multipliers (ADMM) based approach and [280] 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 [190, 393][3] 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(a) and a variant of this setup where labels are also not shared along with raw data is shown in Figure 2(b).



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

Figure 2: 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.

In several settings, the overall communication requirements of split learning and federated learning were compared in [360]. 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 [213, 207], 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 [353].

The values communicated can nevertheless, in general, reveal information about the underlying data. How much, and whether this is acceptable, is likely going to be application and configuration specific. A variation of split learning called NoPeek SplitNN [395] reduces the potential leakage via communicated activations, by reducing their distance correlation [394, 378] 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

[3] See also split learning project website - `https://splitlearning.github.io/`.

enables the split to be made relatively early-on given the decorrelation it provides. 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.

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 [304, 327], 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 Following Hsieh et al. [205], 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.[4] 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.

[4] 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 shift): 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. [289]) 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 (Section 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 Section 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. [151] 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 Section 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. [404].

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 Section 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. [151] 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 — 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 (Section 4.2.1), differential privacy (DP) (Section 4.2.2), and model and update compression (Section 3.5). As noted in Section 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 ap-

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

Table 4: Notation for the discussion of FL algorithms including Federated Averaging.

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**
    $x_{t+1}^i \leftarrow \text{ClientUpdate}(i, x_t)$
  $x_{t+1} \leftarrow \sum_{k=1}^{M} \frac{1}{M} x_{t+1}^i$

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

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

plying updates asynchronously.

One of the most common approaches to optimization for federated learning is the Federated Averaging algorithm [289], an adaption of local-update or parallel SGD.[5] 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 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 Section 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.

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.

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

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. [411, Sec. 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 4 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 gradient, that is, for all choices of x, y

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

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

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

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

$$\mathbb{E}[F(x_T)] - F(x^*) \tag{1}$$

where $x^* = \min_x F(x)$. All convergence rates discussed herein are upper bounds on this term. A summary of convergence results for such functions is given in Table 5.

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 [256, 119, 132]. 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 [370, 271, 428, 399, 334, 318, 233]. The first convergence results for local-update SGD methods were derived under the

Method	Comments	Convergence
Baselines		
mini-batch SGD	batch size KM	$\mathcal{O}\left(\frac{H}{T} + \frac{\sigma}{\sqrt{TKM}}\right)$
SGD	(on 1 worker, no communication)	$\mathcal{O}\left(\frac{H}{TK} + \frac{\sigma}{\sqrt{TK}}\right)$
Baselines with acceleration[a]		
A-mini-batch SGD [256, 119]	batch size KM	$\mathcal{O}\left(\frac{H}{T^2} + \frac{\sigma}{\sqrt{TKM}}\right)$
A-SGD [256]	(on 1 worker, no communication)	$\mathcal{O}\left(\frac{H}{(TK)^2} + \frac{\sigma}{\sqrt{TK}}\right)$
Parallel SGD / Fed-Avg / Local SGD		
Yu et al. [428][b], Stich [370][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 [399][b], Stich and Karimireddy [371]		$\mathcal{O}\left(\frac{HM}{T} + \frac{\sigma}{\sqrt{TKM}}\right)$
Other algorithms		
SCAFFOLD [227]	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] This paper considers 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.

Table 5: Convergence of optimization algorithms in the IID-data setting on N devices, for H-smooth convex functions with variance of the stochastic gradients bounded by σ^2. All convergence rates are upper bounds on Equation (1) after T iterations of the given algorithm (potentially with some iterate averaging scheme). In the IID settings all clients are interchangeable, so without loss of generality we can assume $M = N$.

bounded gradient norm assumption in Stich [370] for strongly-convex and in Yu et al. [428] for non-convex objective functions. These analyses could attain the desired $\sigma/\sqrt{TKM}$ statistical term with suboptimal optimization term (in Table 5 we summarize these results for the middle ground of convex functions).

By removing the bounded gradient assumption, Wang and Joshi [399] and Stich and Karimireddy [371] 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 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. [233] and further in Stich and Karimireddy [371].

For non-convex objectives, Yu et al. [428] 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 [399] 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 [399] 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 [432]. Haddadpour

et al. [191] improves the bounds in Wang and Joshi [399] for functions satisfying the Polyak-Lojasiewicz (PL) condition [226], a generalization of strong convexity. In particular, Haddadpour et al. [191] 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 the error versus iterations convergence, practitioners care the most 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 [400] 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. [289] 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. [411] 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 5 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. [74] 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 either by a central server. 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 [432, 267, 143] 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 Section 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 (Section 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

$$F(x) = \frac{1}{N} \sum_{i=1}^{N} f_i(x) \tag{2}$$

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. [411, Sec. 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, it is generally infeasible in the cross-device setting.

While [370, 428, 399, 371] 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 [266, 261, 265, 401] or the difference between client and global optimum values [264, 232]. Under this assumption, Yu et al. [429] 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 [399]. Li et al. [261] 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. However, it is unclear whether it could provably improve the convergence rate. Khaled et al. [232] 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. [264] 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

Non-IID assumptions

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

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 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. [266]	BCGV	BLGV	Dec; AC; 1step	$O(1/T) + O(1/\sqrt{NT})$
PD-SGD [265]	BCGV	BLGV	Dec; AC	$O(N/T) + O(1/\sqrt{NT})$
MATCHA [401]	BCGV	BLGV	Dec	$O(1/\sqrt{TKM}) + O(M/KT)$
Khaled et al. [232]	BOGV	CVX	AC; LBG	$O(N/T) + O(1/\sqrt{NT})$
Li et al. [264]	BOBD	SCVX; BLGV; BLGN	-	$O(K/T)$
FedProx [261]	BGV	BNCVX	Prox	$O(1/\sqrt{T})$
SCAFFOLD [227]	-	SCVX; BLGV	VR	$O(1/TKM) + O(e^{-T})$

Table 6: The convergence of optimization methods in federated learning with non-IID data assumption. 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.

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 wi-fi) will be selected to participate in the computation. At different times within a day, the clients characteristics can vary significantly. Eichner et al. [151] 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).

We summarize recent theoretical results in Table 6. All the methods in Table 6 assume smoothness or Lipschitz gradients for the local functions on clients. The error bound is measured by optimal objective (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 [264, 227]. Bounded gradient variance, which is a widely used assumption to analyze stochastic gradient methods, is often used when clients use stochastic local updates [266, 264, 265, 401, 227]. Li et al. [264] directly analyzes the Federated Averaging algorithm, which applies K steps of local updates on randomly sampled 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 [399], 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 [399] 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. [265] to the non-IID case. MATCHA [401] 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 and variance reduction techniques For first-order optimization methods, momentum and variance-reduction are promising approaches to improve the optimization and generalization performance. However, there is still no consensus on how to incorporate momentum or variance-reduction techniques into local SGD and Federated Averaging. SCAFFOLD [227] explicitly models the difference in client updates with control variates to perform variance reduction, which could converge rapidly without limiting the difference of clients data distribution. As for momentum schemes, Yu et al. [429] proposed to let each client maintain a local momentum buffer and average these local buffers as well as local model parameters at each communication round. Although this method empirically improves the final accuracy of local SGD, it requires doubled communication cost. Wang et al. [402] proposed another momentum scheme called SlowMo, which can significantly improve the optimization and generalization performance of local SGD without sacrificing throughput. Hsu et al. [206] proposed a momentum scheme similar to SlowMo. While both [429, 402] 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.

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 (Section 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 [392, 54, 431, 22], 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. [196]. Different clients are likely to use language differently, and in fact on-device personalization of model parameters has yielded significant improvements for this problem [403]. 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 [433] immediately become relevant. Notably, Smith et al. [362] 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 a learned clustering or the connected components of a learned graph over the clients [431]. The development of such algorithms is an important open problem. See Section 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 (Section 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 [284, 115, 56], 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 [52], 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 Section 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 [165]. Most notably, the training phase of the popular Reptile algorithm [308] is closely related to Federated Averaging [289] — 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. [234] and Jiang et al. [217] explore the connection between FL and MAML, and show how the MAML setting is a relevant framework to model the personalization objectives for FL. Additional connections with differential privacy were studied in [260].

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 [252, 331] 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), can serve as realistic benchmark problems for MAML algorithms.

- The observed gap between the global and personalized acccuracy [217] 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 Section 6 for a related discussion.

- Jiang et al. [217] 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 [234, 217, 260]. 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 [359]?
- Can non-parameter transfer LTL algorithms, such as ProtoNets [363], 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 [420, 196, 98] 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 [284, 285, 203, 303]. 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 [108] within the federated learning task. Independently of the eventual technical levy of carrying out record linkage privately [348], the task itself happens to be substantially noise prone in the real world [347] and only sparse results have addressed its impact on training models [198]. 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 [320].

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, hperparameter optimization, and debugging) to decentralized datasets and resource-constrained mobile devices. We discuss several of these challanges 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 [339, 237, 241], but it mainly concerns how to improve the model accuracy [59, 364, 321, 159] 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. 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 [381, 75].

3.4.2 Neural Architecture Design

We propose that researchers and engineers explore neural architecture search (NAS) in the federated learning setting. This 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 Section 3.3 still share the same model architecture among all clients. The recent progress in NAS [332, 154, 333, 55, 322, 273, 417, 154, 279] 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 [170], 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. Thus, we believe Neural Architecture Search (NAS) for a global or personalized model in the federated learning setting is a promising direction in research.

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. [32] 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 [32]), 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 [245]. 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 [434, 81, 195, 11, 47, 380]. 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*[6] *compression* – 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[7] – 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) would be realized jointly with (a) and (b) by specific methods.

Much of the existing literature applies to the objective (a) [245, 376, 244, 20, 204]. The impact of (b) on convergence in general has not been studied until very recently; a limited analysis is presented in [231]. A method to address all of (a), (b) and (c) jointly, by constraining the desired model update such that only particular submatrices of model variables are necessary to be available on clients, was proposed by Caldas et al. [87].

In cross-device FL, algorithms generally cannot assume any state is preserved on the clients (Tabel 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 [272, 346, 396, 380, 228, 371] are relevant in this setting, many of which could address both (a) and (b).

[6] In this section, when we refer to "gradient compression" this can be read as also including any delta or update to a model, such as the updates produced by Federated Averaging by taking multiple gradient steps.

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

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 [194, 120, 436, 270, 312], 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 [376] 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 [122]. 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 [412] as well as gradient compression. It is therefore worthwhile to evaluate these data-dependent compression schemes in the federated settings [171].

Compatibility with differential privacy and secure aggregation Many algorithms used in federated learning such as Secure Aggregation [72] and mechanisms of adding noise to achieve differential privacy [7, 290] are not designed to work with compressed or quantized communications. For example, straightforward application of the Secure Aggregation protocol of Bonawitz et al. [73] 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 [75] 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 [371], however those would not give differential privacy, as they break independence between rounds. There is some work on adding discrete noise [13], 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 Section 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 [215, 316, 344]. 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 [8]. 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.

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 [200, 435] 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 [358] 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. [254] 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.

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 1 below).

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 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 moreso 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

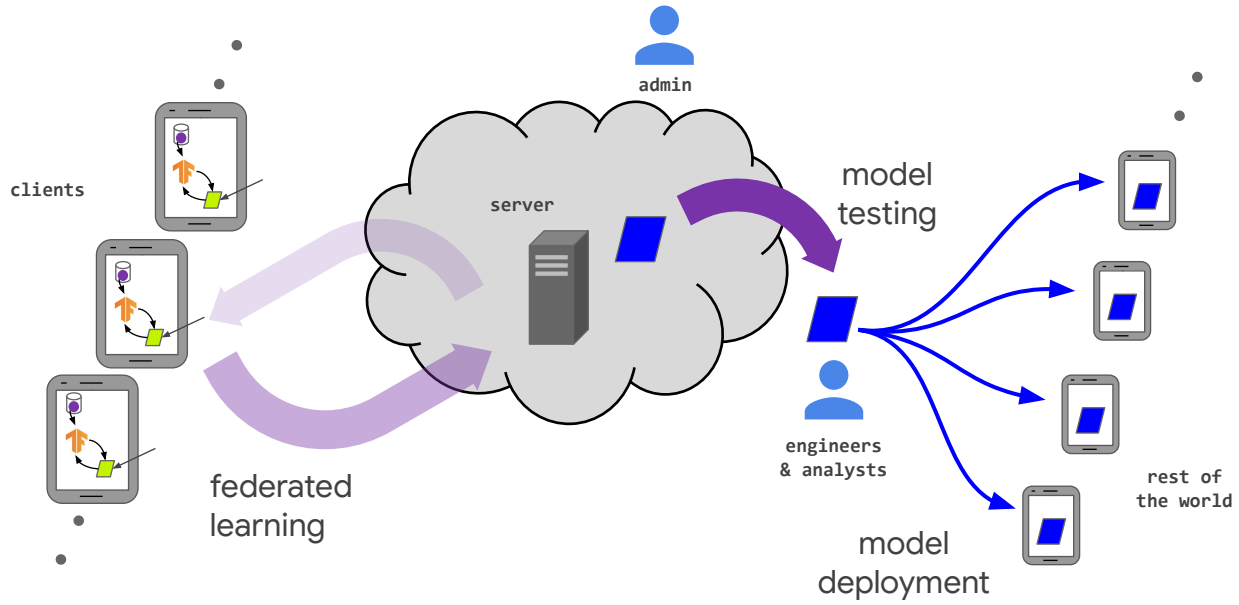


Figure 1: The lifecycle of an FL-trained model and the various actors in a federated learning system. (repeated from Page 7)

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.

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 Section 5.

The remainder of this section is organized as follows. Section 4.1 discusses various threat models against which we wish to give protections. Section 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 Section 4.1. Section 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. Section 4.4 discusses the open problems and challenges in the absence of a fully trusted server. Finally, Section 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 from Section 1, repeated on page 35 for convenience) and their roles to ultimately define relevant threat models (see Table 7). 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. [67, Sec 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[8] 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 $\varepsilon_0 > \varepsilon$.

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 [311].

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 Section 4.2.1.

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

Data/Access Point	Actor	Threat 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 & 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.

Table 7: Various threat models for different adversarial actors.

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 Section 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 Section 4.2.3.

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 [422]. Thanks to both theoretical and engineering breakthroughs, the field has moved from being of a purely theoretical interest to a deployed technology in industry [71, 70, 257, 29, 169, 209, 210]. 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 [211] 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 [172, 14, 182, 77].

It has been known for several decades that any function can be securely computed, even in the presence of malicious adversaries [183]. 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 [309, 172, 302] and neural network training and inference [302, 14, 46]. 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 flavours of HE exist, ranging from general fully homomorphic encryption (FHE) [176] to the more efficient leveled variants [79, 160, 80, 112], for which several implementations exist [3, 350, 4]. 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

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 [373]: • 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.

Table 8: Various technologies along with their characteristics.

in MPC protocols in the cross-silo setting [309, 198]. 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 [345].

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 [102]). 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. [336] and Roth et al. [341] 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 [373]:

- 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 [208, 116], Arm's TrustZone [2, 1], and Sanctum on RISC-V [117], 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 [382] 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 [391].

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 [67]). 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 pairwise additive masking [12, 188, 73], via threshold homomorphic encryption [356, 193, 92], and via generic secure multi-party computation [86]) as well as in the multiple non-colluding servers setting [71, 29, 113]. Secure aggregation can also be approached using trusted execution environments (introduced above), as in [269].

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 [95, 251], often under the heading of mix networks. They have also been studied in the context of trusted computing [67]. Mix networks have found large scale deployment in the form of the Tor network [138].

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 [249], and *information theoretic PIR* (itPIR), in which multiple non-colluding parties are required to execute the server side of the protocol [106].

The main roadblocks to the applicability of PIR have been the following: cPIR has very high computational cost [361], 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 [16, 313, 17, 25, 175]. It has been shown how to construct communication-efficient PIR on a single-server by leveraging side information available at the user [218]. Recent works propose to leverage client local state to speed up PIR. Patel et al. [319] shows how the latter side information can be obtained and a practical hybrid (computational and information theoretic) PIR scheme on a single server was implemented and validated. Corrigan-Gibbs and Kogan [114] present protocols 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 in the past [410] with recent connections to PIR on coded data [139] and communication efficient PIR [66] having been established. 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 [306, 423].

4.2.2 Privacy-Preserving Disclosures

The state-of-the-art model for quantifying and limiting information disclosure about individuals is *differential privacy* (DP) [147, 144, 145], 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':

$$P(\mathcal{A}(D) \in \mathcal{S}) \leq e^{\varepsilon} P(\mathcal{A}(D') \in \mathcal{S}) + \delta. \tag{3}$$

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) [290]. 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 [145], 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 (3) 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) [406, 229]. LDP has been deployed effectively to gather statistics on popular items across large user bases by Google, Apple and Microsoft [156, 135, 136]. It has also been used in federated settings for spam classifier training by Snap [325]. 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 Section 4.4.2, achieving LDP while maintaining utility can be difficult [229, 388]. 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* [146, 356, 67, 105]. 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 saticsfies 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, though we stress that there are many other methods that could be used.

Distributed DP via secure aggregation One promising tool for achieving distributed DP in FL is secure aggregation, discussed above in Section 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) [12, 330, 181, 356, 188].

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 [67] (illustrated in Figure 3). 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 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 messaage 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. [67] 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 [157]. 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 [105, 157, 43, 179, 178].

Hybrid differential privacy Another promising approach is hybrid differential privacy [39], which combines multiple trust models by partitioning users by 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

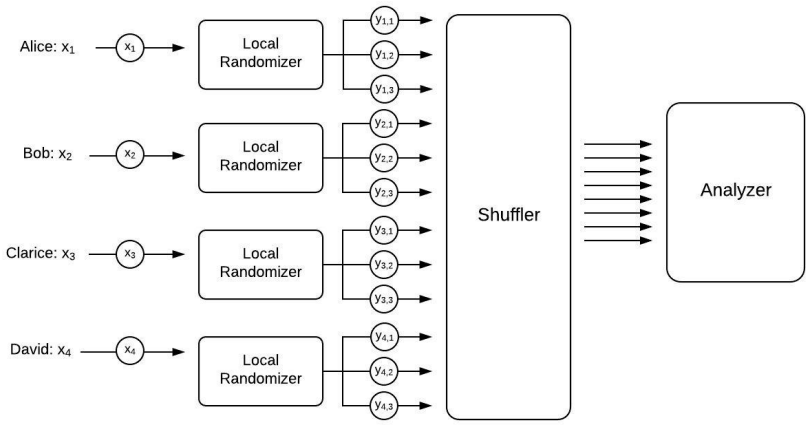


Figure 3: The Encode-Shuffle-Analyze (ESA) framework, illustrated here for 4 players.

model, which typically provides the lowest utility, and conservatively apply it uniformly over the entire user-base. 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 user-base, compared to purely local or central DP mechanisms. For instance, [39] 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 trusted curator model. This enables the design of a mechanism which, in some circumstances, outperforms both the conservative local mechanism applied across all users as well as the trusted curator mechanism applied only across the small fraction of opt-in users. 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. [40] and has been studied under various terms in the literature: checking computations [40], certified computation [295], delegating computations [185], as well as verifiable computing [173].

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. [184]. 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. [317] 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 [349] and Sigma protocols [128] 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 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) [65] 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 [174, 317, 118] and are now available in open-source libraries, such as libsnark [5], and deployed in cryptocurrencies, such as Zcash [57]. 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. [418] 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 [84, 397, 23, 58].

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 [76].

Trusted execution environment and remote attestation We discussed TEEs in Section 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) [373, 385]. 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 attestions [351] and hybrid remote attestation designs [152, 238] 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 [166] 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 7, 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 [167, 357, 91, 293]. 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 [425], 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 endeavour 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 [134]. 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 [7, 290, 288, 62]. 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 [148, 221] or the analytical moments accountant method developed in [7, 297, 299, 405] can be used. The moments accountant method works particularly well with the uniformly subsampled Gaussian mechanism.

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 [24], figuring out how to adaptively bound the contributions of users and clip the model parameters remains an interesting research direction [381, 324]. 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.

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. [163] for convex optimization problems. However, it is unclear whether or not the results in [163] can be carried over to the non-convex setting.

Privacy amplification for non-uniform device sampling procedures The basic update step of cross-device FL operates on a small subset of available clients. When the selected subset of users is uniformly sampled from the underlying population of users, the privacy gains of this sampling procedure can be analyzed using the analytical moments accountant method developed in [405]. However, such a sampling procedure is nearly impossible in practice. This is because (a) the population of devices may not be known to the service provider, and (b) the specific subset of available devices can vary dramatically over time. Thus, quantifying the privacy amplification gains in cross-device FL is an interesting open problem.

Sources of randomness (adapted from [288]) 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 [343].

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 [298]. 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., [296, 192] 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 [137, 275] 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 Section 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 [382] 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 [302, 14]. 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 [337].

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 [147, 148], 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 [125, 124, 89] 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 [145, 9], which can then be used indefinitely without incurring additional privacy loss. This follows from the post-processing guarantees of differential privacy [147]. Augenstein et al. [32] explore demonstrate 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 [124] or privately detecting changes in time series data [125, 89] 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 Section 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 Section 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 [384]. 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 Section 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. [336] for practical settings, that assuming honest (or at least semi-honest) behaviour by the server in a key distribution phase (as done in [73]) 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., [341]) 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" [140]) 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 Section 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 Section 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 [156, 135, 136], a number of results have shown that achieving local differential privacy while preserving utility is challenging particularly, especially in high-dimensional data settings [229, 388, 219, 51, 220, 424, 142, 111]. 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 thus requires a relatively larger user-base or larger choice of ε parameter.

Hybrid differential privacy The hybrid model for differential privacy can help reduce the size of the required user-base by partitioning users based on their trust preferences, but it does not provide privacy amplification of users' locally-added noise. Moreover, it is unclear which application areas and algorithms can best utilize hybrid trust model data [39]. Current work on the hybrid model typically assumes that regardless of the user trust preference, their data comes from the same distribution [39, 141, 53]. 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 privacy amplification from users' locally-added noise, 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 [67] 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 [43]. 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 protocol from [73] has strong privacy guarantees when aggregating client reports. Moreover, the protocol is tailored to the setting of federated learning. For example, it is robust to clients dropping out during the execution (a common feature of cross-device FL) and scales to a large number of parties 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 Section 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., [376]) and communication complexity (see [248] 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 [147, 146].

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 9 and 10 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 [178]).

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 recent work of Agarwal et al. [13] provides one candidate algorithm

Reference	#messages / n	Message size	Expected error
[105]	$\varepsilon\sqrt{n}$	1	$\frac{1}{\varepsilon}\log\frac{n}{\delta}$
[105]	ℓ	1	$\sqrt{n}/\ell + \frac{1}{\varepsilon}\log\frac{1}{\delta}$
[43]	1	$\log n$	$\frac{n^{1/6}\log^{1/3}(1/\delta)}{\varepsilon^{2/3}}$
[180]	$\log(\frac{n}{\varepsilon\delta})$	$\log(\frac{n}{\delta})$	$\frac{1}{\varepsilon}\sqrt{\log\frac{1}{\delta}}$
[44]	$\log(\frac{n}{\delta})$	$\log n$	$\frac{1}{\varepsilon}$
[179] & [45]	$1 + \frac{\log(1/\delta)}{\log n}$	$\log n$	$\frac{1}{\varepsilon}$

Table 9: 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.

	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	[51]	[50]	[406, 157, 43]	[178]	[178]	[291, 369]

Table 10: 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.

$\mathcal{A}$ based on uniform quantization and binomial noise addition. However, it is unknown whether or not the proposed approach is the best one can do in this setting. Therefore, it is of fundamental interest to derive lower 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 Section 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 restored 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 Section 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 [94, 216, 310].

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 Section 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)" [230] 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 2-4am, which is arguably less sensitive.

This approach is formalized in the Pufferfish framework of privacy [235], 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., Section 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 [126, 10]. 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.

5 Robustness to 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 Section 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 [63, 277], model update poisoning [42, 61], and model evasion attacks [377, 63, 186]. 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). 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 [277, 61], 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 Section 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 in Section 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:[9]

1. *untargeted attacks*, or model downgrade attacks, which aim to reduce the model's global accuracy, or "fully break" the global model [63].
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 [100, 277, 42, 61].

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. [42] 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 [42].

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

[9]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 the community to weigh the value of proposed defenses. In Table 11, we propose a few axes of capabilities that are important to consider. We note that this is not a full list. There are many other characteristics of an adversary's capabilities that can be studied.

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 back 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 the federated setting. • *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.
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.

Table 11: Characteristics of an adversary's capabilities in federated settings.

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* [69, 101, 97, 294, 21], *data poisoning* [63, 123, 368, 133], and *evasion* attacks [64, 377, 187, 90, 283]. 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 11 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 [277]. 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 [74]. 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, 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 [255]. 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 [69]. 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 [68, 97, 294, 21, 426, 133].

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 [101, 426], Krum [69], and trimmed mean [426]. Past work has shown that various robust aggregators are provably effective for Byzantine-tolerant distributed learning [372, 69, 101] under appropriate assumptions, even in federated settings [326, 415]. Despite this, Fang et al. [161] 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 [49, 328].

Another line of model update poisoning defenses use redundancy and data shuffling to mitigate Byzantine attacks [97, 328, 129]. 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 [42]. Bhagoji et al. [61] 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 [61, 42].

Existing defenses against backdoor attacks [368, 274, 387, 133, 398, 355, 107] 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 [49]. This paradigm is strongly related to sybil attacks [140], 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 [168]. 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 [63, 100, 239] and untargeted attacks [277, 42].

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. [416], Xie [414] and Xie et al. [415] proposed Byzantine-robust aggregators that successfully defended against label-flipping data poisoning attacks on convolutional neural networks. As discussed in Section 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 Section 5.1.3.

Data sanitization and network pruning Defenses designed specifically for data poisoning attacks frequently rely on "data sanitization" methods [123], which aim to remove poisoned or otherwise anomalous

data. More recent work has developed improved data sanitization methods using robust statistics [368, 355, 387, 133], which often have the benefit of being provably robust to small numbers of outliers [133]. Such methods can be applied to both targeted and untargeted attacks, with some degree of empirical success [355].

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 [274, 398]. Such methods are motivated by previous studies which showed empirically that poisoned data designed to introduce a backdoor often triggers so-called "backdoor neurons" [189]. 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. [240] 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. [61] suggests that data poisoning may be weaker, especially in settings with limited *participation rate* (see Table 11). 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 11). 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 [292]. 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 [64, 377]. 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 [155, 408, 223]. 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 [247, 283]. Such attacks can frequently cause naturally trained models to achieve zero accuracy on image classification benchmarks such as CIFAR-10 or ImageNet [90]. In the black-box setting, models have also been shown to be vulnerable to attacks based on query-access to the model [99, 82] or based on substitute models trained on similar data [377, 315, 386]. 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 [31]. 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 [283, 413, 352]. 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 [155, 383, 354].

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 [212]) 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 [352], 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 [100]. 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 [100, 189] 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 [387, 96, 398, 107]. Difficulties in applying previous defenses (such as those cited above) to the federated setting were discussed in more detail in Section 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) [147], 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 Section 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 Section 4.3.2). This approach has been recently explored by Sun et al. [374], 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. [374] could be extended to include more general adversarial attacks. Therefore, more work remains to verify whether or not DP can indeed be an effective defense. More importantly, it is still unclear how DP's hyberparameters (ℓ_2 norm bound and noise variance) can be generically chosen as a function of the model size/architecture and 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 [281], and in particular in the federated learning context [177]. 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.[10] 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 [258] 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 (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) [73], especially if the clients drop out during the SecAgg protocol. While SecAgg is

[10] 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.

designed to be robust to significant numbers of dropouts [74], 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 [74]. 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 [366]. 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 datacenter 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 [32]. 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 [32]. Developing general-purpose debugging methods for machine learning that do not directly inspect raw data remains a challenge.

Noisy model updates In Section 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 [301] (eg. in vision applications, a client may have a low-resolution camera whose output is scaled to a higher resolution) or noisy labels [307] (eg. 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) 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 [289]) 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 [84]) 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. [415], 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 [326] 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.

6 Ensuring Fairness and Addressing Sources of Bias

Machine learning models can often exhibit surprising and unintended behaviours. When such behaviours 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* [149]. 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 [48, 300] 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 [250].

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 [222], leading to inferior quality predictions for members of these groups (e.g. [85]).

Just as the data access processes used in federated learning may introduce dataset shift and non-independence (Section 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 [263].

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. [150, 338]). For example, suppose training data is based on user interactions with a product which has failed to incorporate inclusive design principle. Then, the user interactions with the product might not express user intents (and hence should be optimized for) 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 [268]), 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 Section 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. 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 accross all individuals during training [199], and via multicalibration, which calibrates for fairness across subsets of the training data [202]. 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 [217, 403]. A similar conception of fairness, as "a more fair distribution of the model performance across devices", is employed in [263].

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 [214, 127], 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 [41, 127, 214, 246] 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 Section 3.3) and "hybrid differential privacy," where some users donate data with lesser privacy guarantees [39].

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 [41]. 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 Section 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 impor-

tant 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 [365].

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 [268, 164, 282, 60, 367]. 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 [83, 93]. 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 [286] and image classification models [85].

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-instutitional 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 Section 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 [190, 198]. 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 [74]. 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. [263]. Frameworks like *agnostic federated learning* [303] provide one approach 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.

In 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 [197, 430, 199]. 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.

7 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 paper 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. 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 [437]. Existing algorithms for solving such problems often do not 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. Moreover, the workshop had a more algorithmic flavor, and so systems-related research topics are somewhat less well represented, despite the fact that building systems for federated learning is a fundamentally important and challenging problem. 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.

References

[1] Android Trusty TEE. https://source.android.com/security/trusty. Accessed: 2019-12-05.

[2] Arm TrustZone Technology. https://developer.arm.com/ip-products/security-ip/trustzone. Accessed: 2019-12-05.

[3] HElib. https://github.com/homenc/HElib, October 2019.

[4] PALISADE lattice cryptography library. https://gitlab.com/palisade/palisade-release, October 2019.

[5] libsnark: a c++ library for zkSNARK proofs. https://github.com/scipr-lab/libsnark, December 2019.

[6] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL https://www.tensorflow.org/. Software available from tensorflow.org.

[7] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 308–318. ACM, 2016.

[8] Omid Abari, Hariharan Rahul, and Dina Katabi. Over-the-air function computation in sensor networks. *CoRR*, abs/1612.02307, 2016. URL http://arxiv.org/abs/1612.02307.

[9] Nazmiye Ceren Abay, Yan Zhou, Murat Kantarcioglu, Bhavani Thuraisingham, and Latanya Sweeney. Privacy preserving synthetic data release using deep learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 510–526. Springer, 2018.

[10] John M Abowd and Ian M Schmutte. An economic analysis of privacy protection and statistical accuracy as social choices. *American Economic Review*, 109(1):171–202, 2019.

[11] Jayadev Acharya, Clement Canonne, and Himanshu Tyagi. Inference under information constraints I: Lower bounds from chi-square contraction. arXiv preprint, arXiv*:1804.06952*, 2018.

[12] Gergely Ács and Claude Castelluccia. I have a DREAM!: DIfferentially PrivatE smart Metering. In *Proceedings of the 13th International Conference on Information Hiding*, IH'11, pages 118–132, Berlin, Heidelberg, 2011. Springer-Verlag. ISBN 978-3-642-24177-2. URL http://dl.acm.org/citation.cfm?id=2042445.2042457.

[13] Naman Agarwal, Ananda Theertha Suresh, Felix X. Yu, Sanjiv Kumar, and Brendan McMahan. cpSGD: Communication-efficient and differentially-private distributed SGD. In *Advances in Neural Information Processing Systems*, pages 7564–7575, 2018.

[14] Nitin Agrawal, Ali Shahin Shamsabadi, Matt J. Kusner, and Adrià Gascón. QUOTIENT: two-party secure neural network training and prediction. In *In Proceedings of the ACM Conference on Computer and Communication Security (CCS)*, 2019.

[15] Rakesh Agrawal and Ramakrishnan Srikant. Privacy-preserving data mining. In *ACM SIGMOD International Conference on Management of Data*, 2000.

[16] Carlos Aguilar-Melchor and Philippe Gaborit. A lattice-based computationally-efficient private information retrieval protocol. *Cryptol. ePrint Arch., Report*, 446, 2007.

[17] Carlos Aguilar-Melchor, Joris Barrier, Laurent Fousse, and Marc-Olivier Killijian. XPIR: Private information retrieval for everyone. *Proceedings on Privacy Enhancing Technologies*, 2016(2):155–174, 2016.

[18] ai.google. Under the hood of the Pixel 2: How AI is supercharging hardware, 2018. URL `https://ai.google/stories/ai-in-hardware/`. Retrieved Nov 2018.

[19] ai.intel. Federated learning for medical imaging, 2019. URL `https://www.intel.ai/federated-learning-for-medical-imaging/`. Retrieved Aug 2019.

[20] Dan Alistarh, Demjan Grubic, Jerry Li, Ryota Tomioka, and Milan Vojnovic. QSGD: Communication-efficient SGD via gradient quantization and encoding. In *NIPS - Advances in Neural Information Processing Systems*, pages 1709–1720, 2017.

[21] Dan Alistarh, Zeyuan Allen-Zhu, and Jerry Li. Byzantine stochastic gradient descent. In *NIPS*, 2018.

[22] Inês Almeida and João Xavier. DJAM: Distributed Jacobi Asynchronous Method for Learning Personal Models. *IEEE Signal Processing Letters*, 25(9):1389–1392, 2018.

[23] Scott Ames, Carmit Hazay, Yuval Ishai, and Muthuramakrishnan 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.

[24] Kareem Amin, Alex Kulesza, Andres Munoz, and Sergei Vassilvtiskii. Bounding user contributions: A bias-variance trade-off in differential privacy. In *International Conference on Machine Learning*, pages 263–271, 2019.

[25] Sebastian Angel, Hao Chen, Kim Laine, and Srinath T. V. Setty. PIR with compressed queries and amortized query processing. In *IEEE Symposium on Security and Privacy*, pages 962–979. IEEE Computer Society, 2018.

[26] George J Annas. HIPAA regulations-a new era of medical-record privacy? *New England Journal of Medicine*, 348(15):1486–1490, 2003.

[27] Apple. Private Federated Learning (NeurIPS 2019 Expo Talk Abstract). `https://nips.cc/ExpoConferences/2019/schedule?talk_id=40`, 2019.

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

[29] Toshinori Araki, Jun Furukawa, Yehuda Lindell, Ariel Nof, and Kazuma 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*, pages 805–817. ACM, 2016.

[30] Mahmoud Assran, Nicolas Loizou, Nicolas Ballas, and Michael Rabbat. Stochastic gradient push for distributed deep learning. In *ICML*, 2019.

[31] Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. *ICML*, 2018.

[32] Sean Augenstein, H. Brendan McMahan, Daniel Ramage, Swaroop Ramaswamy, Peter Kairouz, Mingqing Chen, Rajiv Mathews, and Blaise Aguera y Arcas. Generative models for effective ML on private, decentralized datasets, 2019. URL `https://arxiv.org/abs/1911.06679`.

[33] The Clara Training Framework Authors. NVIDIA Clara, 2019. URL `https://developer.nvidia.com/clara`.

[34] The FATE Authors. Federated AI technology enabler, 2019. URL `https://www.fedai.org/`.

[35] The Leaf Authors. Leaf, 2019. URL https://leaf.cmu.edu/.

[36] The PaddleFL Authors. PaddleFL, 2019. URL https://github.com/PaddlePaddle/PaddleFL.

[37] The PaddlePaddle Authors. PaddlePaddle, 2019. URL http://www.paddlepaddle.org/.

[38] The TFF Authors. TensorFlow Federated, 2019. URL https://www.tensorflow.org/federated.

[39] Brendan Avent, Aleksandra Korolova, David Zeber, Torgeir Hovden, and Benjamin Livshits. BLENDER: Enabling local search with a hybrid differential privacy model. In *26th USENIX Security Symposium (USENIX Security 17)*, pages 747–764, Vancouver, BC, August 2017. USENIX Association. ISBN 978-1-931971-40-9. URL https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/avent.

[40] László Babai, Lance Fortnow, Leonid A. Levin, and Mario Szegedy. Checking computations in polylogarithmic time. In *STOC*, pages 21–31. ACM, 1991.

[41] Eugene Bagdasaryan and Vitaly Shmatikov. Differential privacy has disparate impact on model accuracy. *CoRR*, abs/1905.12101, 2019. URL http://arxiv.org/abs/1905.12101.

[42] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. How to backdoor federated learning. *arXiv preprint arXiv:1807.00459*, 2018.

[43] Borja Balle, James Bell, Adrià Gascón, and Kobbi 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*, pages 638–667, 2019. doi: 10.1007/978-3-030-26951-7_22. URL https://doi.org/10.1007/978-3-030-26951-7_22.

[44] Borja Balle, James Bell, Adria Gascon, and Kobbi Nissim. Differentially private summation with multi-message shuffling. *arXiv preprint arXiv:1906.09116*, 2019.

[45] Borja Balle, James Bell, Adrià Gascón, and Kobbi Nissim. Improved summation from shuffling. *arXiv: 1909.11225*, 2019.

[46] Assi Barak, Daniel Escudero, Anders P. K. Dalskov, and Marcel Keller. Secure evaluation of quantized neural networks. *IACR Cryptology ePrint Archive*, 2019:131, 2019. URL https://eprint.iacr.org/2019/131.

[47] L.P. Barnes, Yanjun Han, and Ayfer Özgür. Lower bounds for learning distributions under communication constraints via Fisher information. arXiv preprint, arXiv*:1902.02890*, 2019.

[48] Solon Barocas, Moritz Hardt, and Arvind Narayanan. *Fairness and Machine Learning*. fairmlbook.org, 2019. http://www.fairmlbook.org.

[49] Moran Baruch, Gilad Baruch, and Yoav Goldberg. A little is enough: Circumventing defenses for distributed learning. *arXiv preprint arXiv:1902.06156*, 2019.

[50] Raef Bassily and Adam Smith. Local, private, efficient protocols for succinct histograms. In *STOC*, pages 127–135, 2015.

[51] Raef Bassily, Uri Stemmer, Abhradeep Guha Thakurta, et al. Practical locally private heavy hitters. In *Advances in Neural Information Processing Systems*, pages 2288–2296, 2017.

[52] Jonathan Baxter. A model of inductive bias learning. *Journal of Artificial Intelligence Research*, 12:149–198, 2000.

[53] Amos Beimel, Aleksandra Korolova, Kobbi Nissim, Or Sheffet, and Uri Stemmer. The power of synergy in differential privacy: Combining a small curator with local randomizers. 2019. Workshop on Privacy Preserving Machine Learning (PriML) at NeurIPS.

[54] Aurélien Bellet, Rachid Guerraoui, Mahsa Taziki, and Marc Tommasi. Personalized and Private Peer-to-Peer Machine Learning. In *AISTATS*, 2018.

[55] Irwan Bello, Barret Zoph, Vijay Vasudevan, and Quoc V Le. Neural optimizer search with reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 459–468. JMLR. org, 2017.

[56] Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. A theory of learning from different domains. *Machine learning*, 79(1-2):151–175, 2010.

[57] Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *IEEE Symposium on Security and Privacy*, pages 459–474. IEEE Computer Society, 2014.

[58] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Scalable zero knowledge with no trusted setup. In *CRYPTO (3)*, volume 11694 of *Lecture Notes in Computer Science*, pages 701–732. Springer, 2019.

[59] James S Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In *Advances in Neural Information Processing Systems*, pages 2546–2554, 2011.

[60] Martín Bertrán, Natalia Martínez, Afroditi Papadaki, Qiang Qiu, Miguel R. D. Rodrigues, Galen Reeves, and Guillermo Sapiro. Learning adversarially fair and transferable representations. In *ICML*, 2019.

[61] Arjun Nitin Bhagoji, Supriyo Chakraborty, Prateek Mittal, and Seraphin Calo. Analyzing federated learning through an adversarial lens. In *Proceedings of the 36th International Conference on Machine Learning*, pages 634–643, 2019.

[62] Abhishek Bhowmick, John Duchi, Julien Freudiger, Gaurav Kapoor, and Ryan Rogers. Protection against reconstruction and its applications in private federated learning. *arXiv preprint arXiv:1812.00984*, 2018.

[63] Battista Biggio, Blaine Nelson, and Pavel Laskov. Poisoning attacks against support vector machines. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, ICML'12, pages 1467–1474, USA, 2012. Omnipress. ISBN 978-1-4503-1285-1. URL `http://dl.acm.org/citation.cfm?id=3042573.3042761`.

[64] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion attacks against machine learning at test time. In *ECML-PKDD*, pages 387–402. Springer, 2013.

[65] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran 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.

[66] R. Bitar and S. E. Rouayheb. Staircase-PIR: Universally robust private information retrieval. In *2018 IEEE Information Theory Workshop (ITW)*, pages 1–5, Nov 2018. doi: 10.1109/ITW.2018.8613532.

[67] Andrea Bittau, Úlfar Erlingsson, Petros Maniatis, Ilya Mironov, Ananth Raghunathan, David Lie, Mitch Rudominer, Ushasree Kode, Julien Tinnes, and Bernhard Seefeld. Prochlo: Strong privacy for analytics in the crowd. In *Proceedings of the 26th Symposium on Operating Systems Principles*, SOSP '17, pages 441–459, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-5085-3. doi: 10.1145/3132747.3132769. URL `http://doi.acm.org/10.1145/3132747.3132769`.

[68] Peva Blanchard, El Mahdi El Mhamdi, Rachid Guerraoui, and Julien Stainer. Machine learning with adversaries: Byzantine tolerant gradient descent. In *Advances in Neural Information Processing Systems*, 2017.

[69] Peva Blanchard, Rachid Guerraoui, Julien Stainer, et al. Machine Learning with Adversaries: Byzantine Tolerant Gradient Descent. In *Advances in Neural Information Processing Systems*, pages 118–128, 2017.

[70] Dan Bogdanov, Riivo Talviste, and Jan Willemson. Deploying secure multi-party computation for financial data analysis - (short paper). In *Financial Cryptography*, volume 7397 of *Lecture Notes in Computer Science*, pages 57–64. Springer, 2012.

[71] Peter Bogetoft, Dan Lund Christensen, Ivan Damgård, Martin Geisler, Thomas P. Jakobsen, Mikkel Krøigaard, Janus Dam Nielsen, Jesper Buus Nielsen, Kurt Nielsen, Jakob Pagter, Michael I. Schwartzbach, and Tomas Toft. Secure multiparty computation goes live. In *Financial Cryptography*, volume 5628 of *Lecture Notes in Computer Science*, pages 325–343. Springer, 2009.

[72] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for federated learning on user-held data. *arXiv preprint arXiv:1611.04482*, 2016.

[73] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for privacy-preserving machine learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1175–1191. ACM, 2017.

[74] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloé M Kiddon, Jakub Konečný, Stefano Mazzocchi, Brendan McMahan, Timon Van Overveldt, David Petrou, Daniel Ramage, and Jason Roselander. Towards federated learning at scale: System design. In *SysML 2019*, 2019. URL `https://arxiv.org/abs/1902.01046`.

[75] Keith Bonawitz, Fariborz Salehi, Jakub Konečný, Brendan McMahan, and Marco Gruteser. Federated learning with autotuned communication-efficient secure aggregation. In *2019 53nd Asilomar Conference on Signals, Systems, and Computers*. IEEE, 2019.

[76] Dan Boneh, Elette Boyle, Henry Corrigan-Gibbs, Niv Gilboa, and Yuval Ishai. Zero-knowledge proofs on secret-shared data via fully linear PCPs. In *CRYPTO (3)*, volume 11694 of *Lecture Notes in Computer Science*, pages 67–97. Springer, 2019.

[77] Florian Bourse, Michele Minelli, Matthias Minihold, and Pascal Paillier. Fast homomorphic evaluation of deep discretized neural networks. In *CRYPTO (3)*, volume 10993 of *Lecture Notes in Computer Science*, pages 483–512. Springer, 2018.

[78] Stephen Boyd, Arpita Ghosh, Balaji Prabhakar, and Devavrat Shah. Randomized gossip algorithms. *IEEE Transactions on Information Theory*, 52(6):2508–2530, 2006.

[79] Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical gapsvp. In *CRYPTO*, volume 7417 of *Lecture Notes in Computer Science*, pages 868–886. Springer, 2012.

[80] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In *ITCS*, pages 309–325. ACM, 2012.

[81] Mark Braverman, Ankit Garg, Tengyu Ma, Huy L. Nguyen, and David 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*, page 1011–1020. ACM, 2016.

[82] Wieland Brendel, Jonas Rauber, and Matthias Bethge. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. *arXiv preprint arXiv:1712.04248*, 2017.

[83] Theodora S Brisimi, Ruidi Chen, Theofanie Mela, Alex Olshevsky, Ioannis Ch Paschalidis, and Wei Shi. Federated learning of predictive models from federated electronic health records. *International journal of medical informatics*, 112:59–67, 2018.

[84] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Gregory 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*.

[85] Joy Buolamwini and Timnit Gebru. Gender shades: Intersectional accuracy disparities in commercial gender classification. In *Conference on fairness, accountability and transparency*, pages 77–91, 2018.

[86] Martin Burkhart, Mario Strasser, Dilip Many, and Xenofontas Dimitropoulos. SEPIA: Privacy-preserving aggregation of multi-domain network events and statistics. *Network*, 1(101101), 2010.

[87] Sebastian Caldas, Jakub Konečný, H Brendan McMahan, and Ameet Talwalkar. Expanding the reach of federated learning by reducing client resource requirements. *arXiv preprint arXiv:1812.07210*, 2018.

[88] Sebastian Caldas, Peter Wu, Tian Li, Jakub Konečný, H Brendan McMahan, Virginia Smith, and Ameet Talwalkar. LEAF: A benchmark for federated settings. *arXiv preprint arXiv:1812.01097*, 2018.

[89] Clément L Canonne, Gautam Kamath, Audra McMillan, Adam Smith, and Jonathan Ullman. The structure of optimal private tests for simple hypotheses. *AarXiv preprint arXiv:1811.11148*, 2019.

[90] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 39–57. IEEE, 2017.

[91] Nicholas Carlini, Chang Liu, Jernej Kos, Úlfar Erlingsson, and Dawn Song. The secret sharer: Measuring unintended neural network memorization & extracting secrets. *arXiv preprint arXiv:1802.08232*, 2018.

[92] T-H Hubert Chan, Elaine Shi, and Dawn Song. Privacy-preserving stream aggregation with fault tolerance. In *International Conference on Financial Cryptography and Data Security*, pages 200–214. Springer, 2012.

[93] Ken Chang, Niranjan Balachandar, Carson Lam, Darvin Yi, James Brown, Andrew Beers, Bruce Rosen, Daniel L Rubin, and Jayashree Kalpathy-Cramer. Distributed deep learning networks among institutions for medical imaging. *Journal of the American Medical Informatics Association*, 25(8):945–954, 2018.

[94] Wei-Ting Chang and Ravi Tandon. On the upload versus download cost for secure and private matrix multiplication. *ArXiv*, abs/1906.10684, 2019.

[95] David Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2), 1981.

[96] Bryant Chen, Wilka Carvalho, Nathalie Baracaldo, Heiko Ludwig, Benjamin Edwards, Taesung Lee, Ian Molloy, and Biplav Srivastava. Detecting backdoor attacks on deep neural networks by activation clustering. *arXiv preprint arXiv:1811.03728*, 2018.

[97] Lingjiao Chen, Hongyi Wang, Zachary B. Charles, and Dimitris S. Papailiopoulos. DRACO: Byzantine-resilient distributed training via redundant gradients. In *Proceedings of the 35th International Conference on Machine Learning, ICML*, 2018.

[98] Mingqing Chen, Rajiv Mathews, Tom Ouyang, and Françoise Beaufays. Federated learning of out-of-vocabulary words. *arXiv preprint 1903.10635*, 2019. URL `http://arxiv.org/abs/1903.10635`.

[99] Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui 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*, pages 15–26. ACM, 2017.

[100] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526*, 2017.

[101] Yudong Chen, Lili Su, and Jiaming Xu. Distributed Statistical Machine Learning in Adversarial Settings: Byzantine Gradient Descent. *POMACS*, 1:44:1–44:25, 2017.

[102] Massimo Chenal and Qiang Tang. On key recovery attacks against existing somewhat homomorphic encryption schemes. In *LATINCRYPT*, volume 8895 of *Lecture Notes in Computer Science*, pages 239–258. Springer, 2014.

[103] Kewei Cheng, Tao Fan, Yilun Jin, Yang Liu, Tianjian Chen, and Qiang Yang. SecureBoost: A lossless federated learning framework. *CoRR*, abs/1901.08755, 2019. URL `http://arxiv.org/abs/1901.08755`.

[104] Raymond Cheng, Fan Zhang, Jernej Kos, Warren He, Nicholas Hynes, Noah Johnson, Ari Juels, Andrew Miller, and Dawn Song. Ekiden: A platform for confidentiality-preserving, trustworthy, and performant smart contracts. In *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 185–200. IEEE, 2019.

[105] Albert Cheu, Adam Smith, Jonathan Ullman, David Zeber, and Maxim Zhilyaev. Distributed differential privacy via shuffling. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 375–403. Springer, 2019.

[106] Benny Chor, Eyal Kushilevitz, Oded Goldreich, and Madhu Sudan. Private information retrieval. *J. ACM*, 45 (6):965–981, November 1998. ISSN 0004-5411. doi: 10.1145/293347.293350. URL `http://doi.acm.org/10.1145/293347.293350`.

[107] Edward Chou, Florian Tramèr, and Giancarlo Pellegrino. SentiNet: Detecting physical attacks against deep learning systems. *arXiv preprint arXiv:1812.00292*, 2018.

[108] P. Christen. *Data matching: concepts and techniques for record linkage, entity resolution, and duplicate detection*. Springer Science & Business Media, 2012.

[109] Gregory Cohen, Saeed Afshar, Jonathan Tapson, and André van Schaik. EMNIST: an extension of MNIST to handwritten letters. *arXiv preprint arXiv:1702.05373*, 2017.

[110] Igor Colin, Aurélien Bellet, Joseph Salmon, and Stéphan Clémençon. Gossip dual averaging for decentralized optimization of pairwise functions. In *ICML*, 2016.

[111] Graham Cormode, Tejas Kulkarni, and Divesh Srivastava. Marginal release under local differential privacy. In *Proceedings of the 2018 International Conference on Management of Data*, pages 131–146. ACM, 2018.

[112] Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. Scale-invariant fully homomorphic encryption over the integers. In *Public Key Cryptography*, volume 8383 of *Lecture Notes in Computer Science*, pages 311–328. Springer, 2014.

[113] Henry Corrigan-Gibbs and Dan Boneh. Prio: Private, robust, and scalable computation of aggregate statistics. In *14th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 17)*, pages 259–282, 2017.

[114] Henry Corrigan-Gibbs and Dmitry Kogan. Private information retrieval with sublinear online time. *IACR Cryptology ePrint Archive*, 2019:1075, 2019.

[115] Corinna Cortes and Mehryar Mohri. Domain adaptation and sample bias correction theory and algorithm for regression. *Theoretical Computer Science*, 519:103–126, 2014.

[116] Victor Costan and Srinivas Devadas. Intel SGX explained. *IACR Cryptology ePrint Archive*, 2016(086):1–118, 2016.

[117] Victor Costan, Ilia Lebedev, and Srinivas Devadas. Sanctum: Minimal hardware extensions for strong software isolation. In *25th {USENIX} Security Symposium ({USENIX} Security 16)*, pages 857–874, 2016.

[118] Craig Costello, Cédric Fournet, Jon Howell, Markulf Kohlweiss, Benjamin Kreuter, Michael Naehrig, Bryan Parno, and Samee Zahur. Geppetto: Versatile verifiable computation. In *IEEE Symposium on Security and Privacy*, pages 253–270. IEEE Computer Society, 2015.

[119] Andrew Cotter, Ohad Shamir, Nati Srebro, and Karthik Sridharan. Better mini-batch algorithms via accelerated gradient methods. In *Advances in Neural Information Processing Systems 24*. 2011.

[120] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. BinaryConnect: Training deep neural networks with binary weights during propagations. In *Advances in neural information processing systems*, pages 3123–3131, 2015.

[121] Pierre Courtiol, Charles Maussion, Matahi Moarii, Elodie Pronier, Samuel Pilcer, Meriem Sefta, Pierre Manceron, Sylvain Toldo, Mikhail Zaslavskiy, Nolwenn Le Stang, et al. Deep learning-based classification of mesothelioma improves prediction of patient outcome. *Nature medicine*, pages 1–7, 2019.

[122] Thomas M Cover and Joy A Thomas. *Elements of information theory*. John Wiley & Sons, 2012.

[123] Gabriela F Cretu, Angelos Stavrou, Michael E Locasto, Salvatore J Stolfo, and Angelos D Keromytis. Casting out demons: Sanitizing training data for anomaly sensors. In *2008 IEEE Symposium on Security and Privacy (sp 2008)*, pages 81–95. IEEE, 2008.

[124] Rachel Cummings, Sara Krehbiel, Kevin Lai, and Uthaipon Tantitongpipat. Differential privacy for growing databases. In *Advances in Neural Information Processing Systems 31*, NeurIPS '18, pages 8864–8873, 2018.

[125] Rachel Cummings, Sara Krehbiel, Yajun Mei, Rui Tuo, and Wanrong Zhang. Differentially private change-point detection. In *Advances in Neural Information Processing Systems 31*, NeurIPS '18, pages 10825–10834, 2018.

[126] Rachel Cummings, Inbal Dekel, Ori Heffetz, and Katrina Ligett. Bringing differential privacy into the experimental economics lab: Theory and an application to a public-good game. Working paper, 2019.

[127] Rachel Cummings, Varun Gupta, Dhamma Kimpara, and Jamie Morgenstern. On the compatibility of privacy and fairness. In *Proceedings of Fairness in User Modeling, Adaptation and Personalization*, FairUMAP, 2019.

[128] Damgård. On σ protocols. `http://www.cs.au.dk/~ivan/Sigma.pdf`.

[129] Deepesh Data, Linqi Song, and Suhas Diggavi. Data encoding for Byzantine-resilient distributed optimization. *arXiv preprint arXiv:1907.02664*, 2019.

[130] Walter de Brouwer. The federated future is ready for shipping. `https://doc.ai/blog/federated-future-ready-shipping/`, March 2019.

[131] Jeffrey Dean, Greg S. Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Quoc V. Le, Mark Z. Mao, Marc'Aurelio Ranzato, Andrew Senior, Paul Tucker, Ke Yang, and Andrew Y. Ng. Large scale distributed deep networks. In *Proceedings of the International Conference on Neural Information Processing Systems*, pages 1223–1231, 2012.

[132] Ofer Dekel, Ran Gilad-Bachrach, Ohad Shamir, and Lin Xiao. Optimal distributed online prediction using mini-batches. *J. Mach. Learn. Res.*, 13(1), January 2012.

[133] Ilias Diakonikolas, Gautam Kamath, Daniel Kane, Jerry Li, Jacob Steinhardt, and Alistair Stewart. Sever: A robust meta-algorithm for stochastic optimization. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 1596–1606, Long Beach, California, USA, 09–15 Jun 2019. PMLR. URL `http://proceedings.mlr.press/v97/diakonikolas19a.html`.

[134] Mario Diaz, Peter Kairouz, Jiachun Liao, and Lalitha Sankar. Theoretical guarantees for model auditing with finite adversaries. *arXiv preprint arXiv:1911.03405*, 2019.

[135] Differential Privacy Team. Learning with privacy at scale. *Apple Machine Learning Journal*, 1(8), 2017. URL `https://machinelearning.apple.com/2017/12/06/learning-with-privacy-at-scale.html`.

[136] Bolin Ding, Janardhan Kulkarni, and Sergey Yekhanin. Collecting telemetry data privately. In *Advances in Neural Information Processing Systems 30*, December 2017. URL `https://www.microsoft.com/en-us/research/publication/collecting-telemetry-data-privately/`.

[137] Zeyu Ding, Yuxin Wang, Guanhong Wang, Danfeng Zhang, and Daniel Kifer. Detecting violations of differential privacy. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, CCS '18, pages 475–489, New York, NY, USA, 2018. ACM. ISBN 978-1-4503-5693-0. doi: 10.1145/3243734.3243818. URL http://doi.acm.org/10.1145/3243734.3243818.

[138] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. Technical report, Naval Research Lab Washington DC, 2004.

[139] Rafael 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)*, pages 1744–1748, June 2018. doi: 10.1109/ISIT.2018.8437805.

[140] John R. Douceur. The sybil attack. In *Revised Papers from the First International Workshop on Peer-to-Peer Systems*, IPTPS '01, pages 251–260, London, UK, UK, 2002. Springer-Verlag. ISBN 3-540-44179-4. URL http://dl.acm.org/citation.cfm?id=646334.687813.

[141] Yatharth Dubey and Aleksandra Korolova. The power of the hybrid model for mean estimation. *arXiv*, December 2018. https://arxiv.org/abs/1811.12040, Workshop on Privacy Preserving Machine Learning at NeurIPS.

[142] John C Duchi, Michael I Jordan, and Martin J Wainwright. Local privacy and statistical minimax rates. In *Foundations of Computer Science (FOCS), 2013 IEEE 54th Annual Symposium on*, pages 429–438. IEEE, 2013.

[143] Sanghamitra Dutta, Gauri Joshi, Soumyadip Ghosh, Parijat Dube, and Priya Nagpurkar. Slow and Stale Gradients Can Win the Race: Error-Runtime Trade-offs in Distributed SGD. *International Conference on Artificial Intelligence and Statistics (AISTATS)*, April 2018. URL https://arxiv.org/abs/1803.01113.

[144] Cynthia Dwork. Differential privacy: A survey of results. In *International Conference on Theory and Applications of Models of Computation*, pages 1–19. Springer, 2008.

[145] Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3–4):211–407, 2014.

[146] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. Our data, ourselves: Privacy via distributed noise generation. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 486–503. Springer, 2006.

[147] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam D. Smith. Calibrating noise to sensitivity in private data analysis. In *IACR Theory of Cryptography Conference (TCC), New York, New York*, volume 3876 of *Lecture Notes in Computer Science*, pages 265–284. Springer-Verlag, 2006. doi: 10.1007/11681878_14.

[148] Cynthia Dwork, Guy N. Rothblum, and Salil Vadhan. Boosting and differential privacy. In *Proceedings of the IEEE 51st Annual Symposium on Foundations of Computer Science*, FOCS '10, pages 51–60, 2010.

[149] Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard Zemel. Fairness through awareness. In *Proceedings of the 3rd innovations in theoretical computer science conference*, pages 214–226. ACM, 2012.

[150] Laurel Eckhouse, Kristian Lum, Cynthia Conti-Cook, and Julie Ciccolini. Layers of bias: A unified approach for understanding problems with risk assessment. *Criminal Justice and Behavior*, 46(2):185–209, 2019.

[151] Hubert Eichner, Tomer Koren, H. Brendan McMahan, Nathan Srebro, and Kunal Talwar. Semi-cyclic stochastic gradient descent. In *Accepted to ICML 2019.*, 2019. URL https://arxiv.org/abs/1904.10120.

[152] Karim Eldefrawy, Gene Tsudik, Aurélien Francillon, and Daniele Perito. SMART: secure and minimal architecture for (establishing dynamic) root of trust. In *NDSS*. The Internet Society, 2012.

[153] Anis Elgabli, Jihong Park, Amrit S Bedi, Mehdi Bennis, and Vaneet Aggarwal. GADMM: Fast and communication efficient framework for distributed machine learning. *arXiv preprint arXiv:1909.00047*, 2019.

[154] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Efficient multi-objective neural architecture search via Lamarckian evolution. *arXiv preprint arXiv:1804.09081*, 2018.

[155] Logan Engstrom, Brandon Tran, Dimitris Tsipras, Ludwig Schmidt, and Aleksander Madry. A rotation and a translation suffice: Fooling CNNs with simple transformations. *arXiv preprint arXiv:1712.02779*, 2017.

[156] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. RAPPOR: Randomized aggregatable privacy-preserving ordinal response. In *ACM CCS*, 2014. ISBN 978-1-4503-2957-6. doi: 10.1145/2660267.2660348. URL http://doi.acm.org/10.1145/2660267.2660348.

[157] Úlfar Erlingsson, Vitaly Feldman, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Abhradeep Thakurta. Amplification by shuffling: From local to central differential privacy via anonymity. In *SODA*, pages 2468–2479, 2019.

[158] EU CORDIS. Machine learning ledger orchestration for drug discovery, 2019. URL 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.

[159] Stefan Falkner, Aaron Klein, and Frank Hutter. BOHB: Robust and efficient hyperparameter optimization at scale. *arXiv preprint arXiv:1807.01774*, 2018.

[160] Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. *IACR Cryptology ePrint Archive*, 2012:144, 2012.

[161] Minghong Fang, Xiaoyu Cao, Jinyuan Jia, and Neil Zhenqiang Gong. Local model poisoning attacks to Byzantine-robust federated learning. *arXiv preprint arXiv:1911.11815*, 2019.

[162] FeatureCloud. FeatureCloud: Our vision, 2019. URL https://featurecloud.eu/about/our-vision/. Retrieved Aug 2019.

[163] Vitaly Feldman, Ilya Mironov, Kunal Talwar, and Abhradeep Thakurta. Privacy amplification by iteration. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 521–532. IEEE, 2018.

[164] Clément Feutry, Pablo Piantanida, Yoshua Bengio, and Pierre Duhamel. Learning anonymized representations with adversarial neural networks. *CoRR*, abs/1802.09386, 2018. URL http://arxiv.org/abs/1802.09386.

[165] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, 2017.

[166] Aurélien Francillon, Quan Nguyen, Kasper Bonne Rasmussen, and Gene Tsudik. A minimalist approach to remote attestation. In *DATE*, pages 1–6. European Design and Automation Association, 2014.

[167] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 1322–1333. ACM, 2015.

[168] Clement Fung, Chris JM Yoon, and Ivan Beschastnikh. Mitigating sybils in federated learning poisoning. *arXiv preprint arXiv:1808.04866*, 2018.

[169] Jun Furukawa, Yehuda Lindell, Ariel Nof, and Or Weinstein. High-throughput secure three-party computation for malicious adversaries and an honest majority. In *EUROCRYPT (2)*, volume 10211 of *Lecture Notes in Computer Science*, pages 225–255, 2017.

[170] Adam Gaier and David Ha. Weight agnostic neural networks. *arXiv preprint arXiv:1906.04358*, 2019.

[171] Venkata Gandikota, Raj Kumar Maity, and Arya Mazumdar. vqSGD: Vector quantized stochastic gradient descent. *arXiv preprint arXiv:1911.07971*, 2019.

[172] Adrià Gascón, Phillipp Schoppmann, Borja Balle, Mariana Raykova, Jack Doerner, Samee Zahur, and David Evans. Privacy-preserving distributed linear regression on high-dimensional data. *PoPETs*, 2017(4):345–364, 2017.

[173] Rosario Gennaro, Craig Gentry, and Bryan Parno. Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In *CRYPTO*, volume 6223 of *Lecture Notes in Computer Science*, pages 465–482. Springer, 2010.

[174] Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic span programs and succinct NIZKs without PCPs. In *EUROCRYPT*, volume 7881 of *Lecture Notes in Computer Science*, pages 626–645. Springer, 2013.

[175] Craig Gentry and Shai Halevi. Compressible FHE with applications to PIR. In *TCC (2)*, volume 11892 of *Lecture Notes in Computer Science*, pages 438–464. Springer, 2019.

[176] Craig Gentry et al. Fully homomorphic encryption using ideal lattices. In *Stoc*, volume 9, pages 169–178, 2009.

[177] Robin C. Geyer, Tassilo Klein, and Moin Nabi. Differentially private federated learning: A client level perspective. *CoRR*, abs/1712.07557, 2017. URL `http://arxiv.org/abs/1712.07557`.

[178] Badih Ghazi, Noah Golowich, Ravi Kumar, Rasmus Pagh, and Ameya Velingker. On the power of multiple anonymous messages. *arXiv:1908.11358*, 2019.

[179] Badih Ghazi, Pasin Manurangsi, Rasmus Pagh, and Ameya Velingker. Private aggregation from fewer anonymous messages. *arXiv:1909.11073*, 2019.

[180] Badih Ghazi, Rasmus Pagh, and Ameya Velingker. Scalable and differentially private distributed aggregation in the shuffled model. *arXiv preprint arXiv:1906.08320*, 2019.

[181] Arpita Ghosh, Tim Roughgarden, and Mukund Sundararajan. Universally utility-maximizing privacy mechanisms. In *Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing*, STOC '09, pages 351–360, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-506-2. doi: 10.1145/1536414.1536464. URL `http://doi.acm.org/10.1145/1536414.1536464`.

[182] Ran Gilad-Bachrach, Nathan Dowlin, Kim Laine, Kristin E. Lauter, Michael Naehrig, and John Wernsing. CryptoNets: Applying neural networks to encrypted data with high throughput and accuracy. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, pages 201–210, 2016. URL `http://proceedings.mlr.press/v48/gilad-bachrach16.html`.

[183] 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, pages 218–229, New York, NY, USA, 1987. ACM. ISBN 0-89791-221-7. doi: 10.1145/28395.28420. URL `http://doi.acm.org/10.1145/28395.28420`.

[184] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, 18(1):186–208, 1989.

[185] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. Delegating computation: interactive proofs for muggles. In *STOC*, pages 113–122. ACM, 2008.

[186] Ian J. Goodfellow, Jonathon Shlens, and Christian 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. URL `http://arxiv.org/abs/1412.6572`.

[187] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *ICLR*, 2015.

[188] Slawomir Goryczka and Li Xiong. A comprehensive comparison of multiparty secure additions with differential privacy. *IEEE Trans. Dependable Sec. Comput.*, 14(5):463–477, 2017. doi: 10.1109/TDSC.2015.2484326. URL https://doi.org/10.1109/TDSC.2015.2484326.

[189] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. BadNets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733*, 2017.

[190] Otkrist Gupta and Ramesh Raskar. Distributed learning of deep neural network over multiple agents. *Journal of Network and Computer Applications*, 116:1–8, 2018.

[191] Farzin Haddadpour, Mohammad Mahdi Kamani, Mehrdad Mahdavi, and Viveck R Cadambe. Local SGD with periodic averaging: Tighter analysis and adaptive synchronization. *arXiv preprint arXiv:1910.13598*, 2019.

[192] Andreas Haeberlen, Benjamin C Pierce, and Arjun Narayan. Differential privacy under fire. In *USENIX Security Symposium*, 2011.

[193] Shai Halevi, Yehuda Lindell, and Benny Pinkas. Secure computation on the web: Computing without simultaneous interaction. In *Annual Cryptology Conference*, pages 132–150. Springer, 2011.

[194] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

[195] Yanjun Han, Ayfer Özgür, and Tsachy Weissman. Geometric lower bounds for distributed parameter estimation under communication constraints. In *Proceedings of Machine Learning Research*, pages 1–26, 75, 2018.

[196] Andrew Hard, Kanishka Rao, Rajiv Mathews, Françoise Beaufays, Sean Augenstein, Hubert Eichner, Chloé Kiddon, and Daniel Ramage. Federated learning for mobile keyboard prediction. *arXiv preprint 1811.03604*, 2018.

[197] Moritz Hardt, Eric Price, and Nathan Srebro. Equality of opportunity in supervised learning. In *Advances in Neural Information Processing Systems*, 2016.

[198] Stephen Hardy, Wilko Henecka, Hamish Ivey-Law, Richard Nock, Giorgio Patrini, Guillaume Smith, and Brian Thorne. Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption. November 2017.

[199] Tatsunori Hashimoto, Megha Srivastava, Hongseok Namkoong, and Percy Liang. Fairness without demographics in repeated loss minimization. In *International Conference on Machine Learning*, pages 1934–1943, 2018.

[200] Chaoyang He, Conghui Tan, Hanlin Tang, Shuang Qiu, and Ji Liu. Central server free federated learning over single-sided trust social networks. *arXiv preprint arXiv:1910.04956*, 2019.

[201] Lie He, An Bian, and Martin Jaggi. COLA: Decentralized linear learning. In *NeurIPS 2018 - Advances in Neural Information Processing Systems 31*, 2018.

[202] Úrsula Hébert-Johnson, Michael Kim, Omer Reingold, and Guy Rothblum. Multicalibration: Calibration for the (computationally-identifiable) masses. In *International Conference on Machine Learning*, pages 1944–1953, 2018.

[203] Judy Hoffman, Mehryar Mohri, and Ningshan Zhang. Algorithms and theory for multiple-source adaptation. In *Advances in Neural Information Processing Systems*, pages 8246–8256, 2018.

[204] Samuel Horvath, Chen-Yu Ho, Ludovit Horvath, Atal Narayan Sahu, Marco Canini, and Peter Richtarik. Natural compression for distributed deep learning. *arXiv preprint arXiv:1905.10988*, 2019.

[205] Kevin Hsieh, Amar Phanishayee, Onur Mutlu, and Phillip B. Gibbons. The non-IID data quagmire of decentralized machine learning, 2019. URL https://arxiv.org/abs/1910.00189.

[206] Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown. Measuring the effects of non-identical data distribution for federated visual classification. *arXiv preprint arXiv:1909.06335*, 2019.

[207] Zhouyuan Huo, Bin Gu, and Heng Huang. Training neural networks using features replay. In *Advances in Neural Information Processing Systems*, pages 6659–6668, 2018.

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

[209] Mihaela Ion, Ben Kreuter, Erhan Nergiz, Sarvar Patel, Shobhit Saxena, Karn Seth, David Shanahan, and Moti Yung. Private intersection-sum protocol with applications to attributing aggregate ad conversions. *IACR Cryptology ePrint Archive*, 2017:738, 2017.

[210] Mihaela Ion, Ben Kreuter, Ahmet Erhan Nergiz, Sarvar Patel, Mariana Raykova, Shobhit Saxena, Karn Seth, David Shanahan, and Moti Yung. On deploying secure computing commercially: Private intersection-sum protocols and their business applications. *IACR Cryptology ePrint Archive*, 2019:723, 2019.

[211] Yuval Ishai, Joe Kilian, Kobbi Nissim, and Erez Petrank. Extending oblivious transfers efficiently. In *CRYPTO*, volume 2729 of *Lecture Notes in Computer Science*, pages 145–161. Springer, 2003.

[212] Jörn-Henrik Jacobsen, Jens Behrmannn, Nicholas Carlini, Florian Tramèr, and Nicolas Papernot. Exploiting excessive invariance caused by norm-bounded adversarial robustness. *arXiv preprint arXiv:1903.10484*, 2019.

[213] Max Jaderberg, Wojciech Marian Czarnecki, Simon Osindero, Oriol Vinyals, Alex Graves, David Silver, and Koray Kavukcuoglu. Decoupled neural interfaces using synthetic gradients. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1627–1635. JMLR. org, 2017.

[214] Matthew Jagielski, Michael J. Kearns, Jieming Mao, Alina Oprea, Aaron Roth, Saeed Sharifi-Malvajerdi, and Jonathan Ullman. Differentially private fair learning. *CoRR*, abs/1812.02696, 2018. URL http://arxiv.org/abs/1812.02696.

[215] Eunjeong Jeong, Seungeun Oh, Hyesung Kim, Jihong Park, Mehdi Bennis, and Seong-Lyun Kim. Communication-efficient on-device machine learning: Federated distillation and augmentation under non-IID private data. *CoRR*, abs/1811.11479, 2018. URL http://arxiv.org/abs/1811.11479.

[216] Zhuqing Jia and Syed Ali Jafar. On the capacity of secure distributed matrix multiplication. *ArXiv*, abs/1908.06957, 2019.

[217] Yihan Jiang, Jakub Konečný, Keith Rush, and Sreeram Kannan. Improving federated learning personalization via model agnostic meta learning. *arXiv preprint arXiv:1909.12488*, 2019.

[218] 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)*, pages 1099–1106, Oct 2017. doi: 10.1109/ALLERTON.2017.8262860.

[219] Peter Kairouz, Sewoong Oh, and Pramod Viswanath. Extremal mechanisms for local differential privacy. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2879–2887. Curran Associates, Inc., 2014.

[220] Peter Kairouz, Keith Bonawitz, and Daniel Ramage. Discrete distribution estimation under local privacy. In *International Conference on Machine Learning*, pages 2436–2444, 2016.

[221] Peter Kairouz, Sewoong Oh, and Pramod Viswanath. The composition theorem for differential privacy. *IEEE Transactions on Information Theory*, 63(6):4037–4049, 2017.

[222] Toshihiro Kamishima, Shotaro Akaho, and Jun Sakuma. Fairness-aware learning through regularization approach. In *2011 IEEE 11th International Conference on Data Mining Workshops*, pages 643–650. IEEE, 2011.

[223] Daniel Kang, Yi Sun, Dan Hendrycks, Tom Brown, and Jacob Steinhardt. Testing robustness against unforeseen adversaries. *arXiv preprint arXiv:1908.08016*, 2019.

[224] Jiawen Kang, Zehui Xiong, Dusit Niyato, Shengli Xie, and Junshan Zhang. Incentive mechanism for reliable federated learning: A joint optimization approach to combining reputation and contract theory. *IEEE Internet of Things Journal*, 2019.

[225] Jiawen Kang, Zehui Xiong, Dusit Niyato, Han Yu, Ying-Chang Liang, and Dong In 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*, pages 1–5, 2019.

[226] Hamed Karimi, Julie Nutini, and Mark 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*, pages 795–811. Springer, 2016.

[227] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank J Reddi, Sebastian U Stich, and Ananda Theertha Suresh. SCAFFOLD: Stochastic controlled averaging for on-device federated learning. *arXiv preprint arXiv:1910.06378*, 2019.

[228] Sai Praneeth Karimireddy, Quentin Rebjock, Sebastian Stich, and Martin Jaggi. Error feedback fixes SignSGD and other gradient compression schemes. In *ICML*, 2019.

[229] Shiva Prasad Kasiviswanathan, Homin K. Lee, Kobbi Nissim, Sofya Raskhodnikova, and Adam D. Smith. What can we learn privately? *SIAM J. Comput.*, 40(3):793–826, 2011. URL `https://doi.org/10.1137/090756090`.

[230] Michael J. Kearns, Aaron Roth, Zhiwei Steven Wu, and Grigory Yaroslavtsev. Privacy for the protected (only). *CoRR*, abs/1506.00242, 2015. URL `http://arxiv.org/abs/1506.00242`.

[231] Ahmed Khaled and Peter Richtárik. Gradient descent with compressed iterates. *arXiv preprint arXiv:1909.04716*, 2019.

[232] Ahmed Khaled, Konstantin Mishchenko, and Peter Richtárik. First analysis of local GD on heterogeneous data, 2019. URL `https://arxiv.org/abs/1909.04715`.

[233] Ahmed Khaled, Konstantin Mishchenko, and Peter Richtárik. Better communication complexity for local SGD, 2019. URL `https://arxiv.org/abs/1909.04746`.

[234] Mikhail Khodak, Maria-Florina Balcan, and Ameet Talwalkar. Adaptive gradient-based meta-learning methods. In *Advances in Neural Information Processing Systems*, 2019.

[235] Daniel Kifer and Ashwin Machanavajjhala. Pufferfish: A framework for mathematical privacy definitions. *ACM Transactions on Database Systems*, 39(1):3:1–3:36, 2014.

[236] Yejin Kim, Jimeng Sun, Hwanjo Yu, and Xiaoqian 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*, pages 887–895, 2017. doi: 10.1145/3097983.3098118. URL `https://doi.org/10.1145/3097983.3098118`.

[237] Ross D. King, Cao Feng, and Alistair Sutherland. StatLog: comparison of classification algorithms on large real-world problems. *Applied Artificial Intelligence an International Journal*, 9(3):289–333, 1995.

[238] Patrick Koeberl, Steffen Schulz, Ahmad-Reza Sadeghi, and Vijay Varadharajan. TrustLite: a security architecture for tiny embedded devices. In *EuroSys*, pages 10:1–10:14. ACM, 2014.

[239] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1885–1894. JMLR. org, 2017.

[240] Pang Wei Koh, Jacob Steinhardt, and Percy Liang. Stronger data poisoning attacks break data sanitization defenses. *arXiv preprint arXiv:1811.00741*, 2018.

[241] Ron Kohavi and George H John. Automatic parameter selection by minimizing estimated error. In *Machine Learning Proceedings 1995*, pages 304–312. Elsevier, 1995.

[242] Anastasia Koloskova, Tao Lin, Sebastian U Stich, and Martin Jaggi. Decentralized deep learning with arbitrary communication compression. *arXiv:1907.09356*, 2019.

[243] Anastasia Koloskova, Sebastian U Stich, and Martin Jaggi. Decentralized Stochastic Optimization and Gossip Algorithms with Compressed Communication. In *ICML*, 2019.

[244] Jakub Konečný and Peter Richtárik. Randomized distributed mean estimation: Accuracy vs communication. *Frontiers in Applied Mathematics and Statistics*, 4:62, 2018.

[245] Jakub Konečný, H Brendan McMahan, Felix X. Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

[246] Satya Kuppam, Ryan McKenna, David Pujol, Michael Hay, Ashwin Machanavajjhala, and Gerome Miklau. Fair decision making using privacy-protected data. *CoRR*, abs/1905.12744, 2019. URL `http://arxiv.org/abs/1905.12744`.

[247] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial machine learning at scale. *arXiv preprint arXiv:1611.01236*, 2016.

[248] Eyal Kushilevitz and Noam Nisan. *Communication Complexity*. Cambridge University Press, New York, NY, USA, 1997. ISBN 0-521-56067-5.

[249] Eyal Kushilevitz and Rafail Ostrovsky. Replication is not needed: Single database, computationally-private information retrieval. In *In Proc. of the 38th Annu. IEEE Symp. on Foundations of Computer Science*, pages 364–373, 1997.

[250] Matt J Kusner, Joshua Loftus, Chris Russell, and Ricardo Silva. Counterfactual fairness. In *Advances in Neural Information Processing Systems*, pages 4066–4076, 2017.

[251] Albert Kwon, David Lazar, Srinivas Devadas, and Bryan Ford. Riffle. *Proceedings on Privacy Enhancing Technologies*, 2016(2):115–134, 2016.

[252] Brenden M. Lake, Ruslan Salakhutdinov, Jason Gross, and Joshua B. Tenenbaum. One shot learning of simple visual concepts. In *Proceedings of the Conference of the Cognitive Science Society (CogSci)*, 2017.

[253] Anusha Lalitha, Osman Cihan Kilinc, Tara Javidi, and Farinaz Koushanfar. Peer-to-peer Federated Learning on Graphs. Technical report, arXiv:1901.11173, 2019.

[254] Anusha Lalitha, Xinghan Wang, Osman Kilinc, Yongxi Lu, Tara Javidi, and Farinaz Koushanfar. Decentralized Bayesian learning over graphs. *arXiv preprint: 1905.10466*, 2019.

[255] Leslie Lamport, Robert Shostak, and Marshall Pease. The Byzantine generals problem. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 4(3):382–401, 1982.

[256] Guanghui Lan. An optimal method for stochastic composite optimization. *Mathematical Programming*, 133 (1):365–397, Jun 2012. ISSN 1436-4646. doi: 10.1007/s10107-010-0434-y. URL `https://doi.org/10.1007/s10107-010-0434-y`.

[257] Andrei Lapets, Nikolaj Volgushev, Azer Bestavros, Frederick Jansen, and Mayank Varia. Secure MPC for analytics as a web application. In *SecDev*, pages 73–74. IEEE Computer Society, 2016.

[258] Mathias Lécuyer, Vaggelis Atlidakis, Roxana Geambasu, Daniel Hsu, and Suman 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*, pages 656–672, 2019. doi: 10.1109/SP.2019.00044. URL https://doi.org/10.1109/SP.2019.00044.

[259] David Leroy, Alice Coucke, Thibaut Lavril, Thibault Gisselbrecht, and Joseph Dureau. Federated learning for keyword spotting. *arXiv preprint arXiv:1810.05512*, 2018.

[260] Jeffrey Li, Mikhail Khodak, Sebastian Caldas, and Ameet Talwalkar. Differentially private meta-learning. *arXiv preprint arXiv:1909.05830*, 2019.

[261] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks, 2018. URL https://arxiv.org/abs/1812.06127.

[262] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions, 2019.

[263] Tian Li, Maziar Sanjabi, and Virginia Smith. Fair resource allocation in federated learning. *arXiv preprint arXiv:1905.10497*, 2019.

[264] Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. On the convergence of fedavg on non-IID data. *arXiv preprint arXiv:1907.02189*, 2019.

[265] Xiang Li, Wenhao Yang, Shusen Wang, and Zhihua Zhang. Communication efficient decentralized training with multiple local updates. *arXiv preprint arXiv:1910.09126*, 2019.

[266] Xiangru Lian, Ce Zhang, Huan Zhang, Cho-Jui Hsieh, Wei Zhang, and Ji Liu. Can Decentralized Algorithms Outperform Centralized Algorithms? A Case Study for Decentralized Parallel Stochastic Gradient Descent. In *NIPS*, 2017.

[267] Xiangru Lian, Wei Zhang, Ce Zhang, and Ji Liu. Asynchronous Decentralized Parallel Stochastic Gradient Descent. In *ICML*, 2018.

[268] Jiachun Liao, Chong Huang, Peter Kairouz, and Lalitha Sankar. Learning generative adversarial representations (GAP) under fairness and censoring constraints. *arXiv preprint arXiv:1910.00411*, 2019.

[269] David Lie and Petros Maniatis. Glimmers: Resolving the privacy/trust quagmire. In *Proceedings of the 16th Workshop on Hot Topics in Operating Systems*, pages 94–99. ACM, 2017.

[270] Darryl Lin, Sachin Talathi, and Sreekanth Annapureddy. Fixed point quantization of deep convolutional networks. In *International Conference on Machine Learning*, pages 2849–2858, 2016.

[271] Tao Lin, Sebastian U Stich, and Martin Jaggi. Don't use large mini-batches, use local SGD. *arXiv preprint arXiv:1808.07217*, 2018.

[272] Yujun Lin, Song Han, Huizi Mao, Yu Wang, and William J Dally. Deep gradient compression: Reducing the communication bandwidth for distributed training. *arXiv preprint arXiv:1712.01887*, 2017.

[273] Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018.

[274] Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Fine-pruning: Defending against backdooring attacks on deep neural networks. In *International Symposium on Research in Attacks, Intrusions, and Defenses*, pages 273–294. Springer, 2018.

[275] Xiyang Liu and Sewoong Oh. Minimax rates of estimating approximate differential privacy. *arXiv preprint arXiv:1905.10335*, 2019.

[276] Yang Liu, Tianjian Chen, and Qiang Yang. Secure federated transfer learning. *arXiv preprint arXiv:1812.03337*, 2018.

[277] Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu 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. URL http://wp.internetsociety.org/ndss/wp-content/uploads/sites/25/2018/02/ndss2018_03A-5_Liu_paper.pdf.

[278] Jiahuan Luo, Xueyang Wu, Yun Luo, Anbu Huang, Yunfeng Huang, Yang Liu, and Qiang Yang. Real-world image datasets for federated learning. *arXiv preprint arXiv:1910.11089*, 2019.

[279] Renqian Luo, Fei Tian, Tao Qin, Enhong Chen, and Tie-Yan Liu. Neural architecture optimization. In *Advances in neural information processing systems*, pages 7816–7827, 2018.

[280] Jing Ma, Qiuchen Zhang, Jian Lou, Joyce Ho, Li Xiong, and Xiaoqian Jiang. Privacy-preserving tensor factorization for collaborative health data analysis. In *ACM CIKM*, volume 2, 2019.

[281] Yuzhe Ma, Xiaojin Zhu, and Justin Hsu. Data poisoning against differentially-private learners: Attacks and defenses. In *International Joint Conference on Artificial Intelligence (IJCAI), Macao, China*, 2019. URL https://arxiv.org/abs/1903.09860.

[282] David Madras, Elliot Creager, Toniann Pitassi, and Richard Zemel. Learning adversarially fair and transferable representations. In *ICML*, 2018.

[283] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *ICLR*, 2017.

[284] Yishay Mansour, Mehryar Mohri, and Afshin Rostamizadeh. Domain adaptation: Learning bounds and algorithms. *arXiv preprint arXiv:0902.3430*, 2009.

[285] Yishay Mansour, Mehryar Mohri, and Afshin Rostamizadeh. Domain adaptation with multiple sources. In *Advances in neural information processing systems*, pages 1041–1048, 2009.

[286] Alicia R Martin, Masahiro Kanai, Yoichiro Kamatani, Yukinori Okada, Benjamin M Neale, and Mark J Daly. Current clinical use of polygenic scores will risk exacerbating health disparities. *BioRxiv*, page 441261, 2019.

[287] H Brendan McMahan and Daniel Ramage. Federated learning: Collaborative machine learning without centralized training data, April 2017. URL https://ai.googleblog.com/2017/04/federated-learning-collaborative.html. Google AI Blog.

[288] H Brendan McMahan, Galen Andrew, Ulfar Erlingsson, Steve Chien, Ilya Mironov, Nicolas Papernot, and Peter Kairouz. A general approach to adding differential privacy to iterative training procedures. dec 2018. *URL https://arxiv. org/abs*, 1812.

[289] H Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, pages 1273–1282, 2017 (original version on arxiv Feb. 2016).

[290] H Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. Learning differentially private recurrent language models. In *International Conference on Learning Representations (ICLR)*, 2018.

[291] Frank McSherry and Kunal Talwar. Mechanism design via differential privacy. In *FOCS*, pages 94–103, 2007.

[292] Shike Mei and Xiaojin Zhu. Using machine teaching to identify optimal training-set attacks on machine learners. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[293] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. Exploiting unintended feature leakage in collaborative learning. *arXiv preprint arXiv:1805.04049*, 2018.

[294] El Mahdi El Mhamdi, Rachid Guerraoui, and Sébastien Rouault. The hidden vulnerability of distributed learning in Byzantium. In *ICML*, 2018.

[295] Silvio Micali. Computationally sound proofs. *SIAM J. Comput.*, 30(4):1253–1298, 2000.

[296] Ilya Mironov. On significance of the least significant bits for differential privacy. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 650–661. ACM, 2012.

[297] Ilya Mironov. Rényi differential privacy. In *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*, pages 263–275. IEEE, 2017.

[298] Ilya Mironov, Omkant Pandey, Omer Reingold, and Salil Vadhan. Computational differential privacy. In *Advances in Cryptology—CRYPTO*, pages 126–142, 2009.

[299] Ilya Mironov, Kunal Talwar, and Li Zhang. R\'enyi differential privacy of the sampled Gaussian mechanism. *arXiv preprint arXiv:1908.10530*, 2019.

[300] Shira Mitchell, Eric Potash, and Solon Barocas. Prediction-based decisions and fairness: A catalogue of choices, assumptions, and definitions. *arXiv preprint arXiv:1811.07867*, 2018.

[301] Volodymyr Mnih and Geoffrey E Hinton. Learning to label aerial images from noisy data. In *Proceedings of the 29th International conference on machine learning (ICML-12)*, pages 567–574, 2012.

[302] Payman Mohassel and Yupeng Zhang. SecureML: A system for scalable privacy-preserving machine learning. In *IEEE Symposium on Security and Privacy*, pages 19–38. IEEE Computer Society, 2017.

[303] Mehryar Mohri, Gary Sivek, and Ananda Theertha Suresh. Agnostic Federated Learning. In *ICML*, 2019.

[304] Jose G. Moreno-Torres, Troy Raeder, RocíO Alaiz-RodríGuez, Nitesh V. Chawla, and Francisco Herrera. A unifying view on dataset shift in classification. *Pattern Recogn.*, 45(1), January 2012.

[305] Musketeer. Musketeer: About, 2019. URL `http://musketeer.eu/project/`. Retrieved Aug 2019.

[306] Carolina Naim, Fangwei Ye, and Salim El Rouayheb. ON-OFF privacy with correlated requests. In *2019 IEEE International Symposium on Information Theory (ISIT)*, July 2019.

[307] Nagarajan Natarajan, Inderjit S Dhillon, Pradeep K Ravikumar, and Ambuj Tewari. Learning with noisy labels. In *Advances in neural information processing systems*, pages 1196–1204, 2013.

[308] Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018.

[309] Valeria Nikolaenko, Udi Weinsberg, Stratis Ioannidis, Marc Joye, Dan Boneh, and Nina Taft. Privacy-preserving ridge regression on hundreds of millions of records. In *IEEE Symposium on Security and Privacy*, pages 334–348. IEEE Computer Society, 2013.

[310] Chaoyue Niu, Fan Wu, Shaojie Tang, Lifeng Hua, Rongfei Jia, Chengfei Lv, Zhihua Wu, and Guihai Chen. Secure federated submodel learning. *arXiv preprint arXiv:1911.02254*, 2019.

[311] NSA. Defense in depth: A practical strategy for achieving Information Assurance in today's highly networked environments. 2012.

[312] Deniz Oktay, Johannes Ballé, Saurabh Singh, and Abhinav Shrivastava. Model compression by entropy penalized reparameterization. *arXiv preprint arXiv:1906.06624*, 2019.

[313] Femi Olumofin and Ian Goldberg. Revisiting the computational practicality of private information retrieval. In *International Conference on Financial Cryptography and Data Security*, pages 158–172. Springer, 2011.

[314] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010.

[315] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia conference on computer and communications security*, pages 506–519. ACM, 2017.

[316] Jihong Park, Sumudu Samarakoon, Mehdi Bennis, and Mérouane Debbah. Wireless network intelligence at the edge. *CoRR*, abs/1812.02858, 2018. URL http://arxiv.org/abs/1812.02858.

[317] Bryan Parno, Jon Howell, Craig Gentry, and Mariana Raykova. Pinocchio: nearly practical verifiable computation. *Commun. ACM*, 59(2):103–112, 2016.

[318] Kumar Kshitij Patel and Aymeric Dieuleveut. Communication trade-offs for synchronized distributed SGD with large step size. *NeurIPS*, 2019.

[319] Sarvar Patel, Giuseppe Persiano, and Kevin Yeo. Private stateful information retrieval. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, CCS '18, pages 1002–1019, New York, NY, USA, 2018. ACM. ISBN 978-1-4503-5693-0. doi: 10.1145/3243734.3243821. URL http://doi.acm.org/10.1145/3243734.3243821.

[320] Giorgio Patrini, Richard Nock, Stephen Hardy, and Tibério 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*, pages 1909–1917, 2016. URL http://www.ijcai.org/Abstract/16/273.

[321] Fabian Pedregosa. Hyperparameter optimization with approximate gradient. *arXiv preprint arXiv:1602.02355*, 2016.

[322] Hieu Pham, Melody Guan, Barret Zoph, Quoc Le, and Jeff Dean. Efficient neural architecture search via parameter sharing. In *International Conference on Machine Learning*, pages 4092–4101, 2018.

[323] Sundar Pichai. Google's Sundar Pichai: Privacy Should Not Be a Luxury Good. *New York Times*, May 7, 2019.

[324] Venkatadheeraj Pichapati, Ananda Theertha Suresh, Felix X Yu, Sashank J Reddi, and Sanjiv Kumar. AdaCliP: Adaptive clipping for private SGD. *arXiv preprint arXiv:1908.07643*, 2019.

[325] Vasyl Pihur, Aleksandra Korolova, Frederick Liu, Subhash Sankuratripati, Moti Yung, Dachuan Huang, and Ruogu Zeng. Differentially-private "Draw and Discard" machine learning. *CoRR*, abs/1807.04369, 2018. URL http://arxiv.org/abs/1807.04369.

[326] Krishna Pillutla, Sham M. Kakade, and Zaid Harchaoui. Robust aggregation for federated learning. Technical report, 2019. URL https://krishnap25.github.io/papers/2019_rfa.pdf.

[327] Joaquin Quionero-Candela, Masashi Sugiyama, Anton Schwaighofer, and Neil D. Lawrence. *Dataset Shift in Machine Learning*. The MIT Press, 2009. ISBN 0262170051, 9780262170055.

[328] Shashank Rajput, Hongyi Wang, Zachary Charles, and Dimitris Papailiopoulos. DETOX: A redundancy-based framework for faster and more robust gradient aggregation. *arXiv preprint arXiv:1907.12205*, 2019.

[329] Swaroop Ramaswamy, Rajiv Mathews, Kanishka Rao, and Françoise Beaufays. Federated learning for emoji prediction in a mobile keyboard. *arXiv preprint 1906.04329*, 2019.

[330] Vibhor Rastogi and Suman 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, pages 735–746, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0032-2. doi: 10.1145/1807167.1807247. URL http://doi.acm.org/10.1145/1807167.1807247.

[331] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In *Proceedings of the 5th International Conference on Learning Representations*, 2017.

[332] Esteban Real, Sherry Moore, Andrew Selle, Saurabh Saxena, Yutaka Leon Suematsu, Jie Tan, Quoc V Le, and Alexey Kurakin. Large-scale evolution of image classifiers. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2902–2911. JMLR. org, 2017.

[333] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4780–4789, 2019.

[334] Amirhossein Reisizadeh, Aryan Mokhtari, Hamed Hassani, Ali Jadbabaie, and Ramtin Pedarsani. Fedpaq: A communication-efficient federated learning method with periodic averaging and quantization. *arXiv preprint arXiv:1909.13014*, 2019.

[335] Amirhossein Reisizadeh, Hossein Taheri, Aryan Mokhtari, Hamed Hassani, and Ramtin Pedarsani. Robust and communication-efficient collaborative learning. *arXiv:1907.10595*, 2019.

[336] Leonid Reyzin, Adam D. Smith, and Sophia Yakoubov. Turning HATE into LOVE: homomorphic ad hoc threshold encryption for scalable MPC. *IACR Cryptology ePrint Archive*, 2018:997, 2018.

[337] M Sadegh Riazi, Kim Laine, Blake Pelton, and Wei Dai. HEAX: High-performance architecture for computation on homomorphically encrypted data in the cloud. *arXiv preprint arXiv:1909.09731*, 2019.

[338] Rashida Richardson, Jason Schultz, and Kate 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.

[339] Brian D Ripley. Statistical aspects of neural networks. *Networks and chaos—statistical and probabilistic aspects*, 50:40–123, 1993.

[340] Ronald L Rivest, Len Adleman, and Michael L Dertouzos. On data banks and privacy homomorphisms. *Foundations of Secure Computation, Academia Press*, pages 169–179, 1978.

[341] Edo Roth, Daniel Noble, Brett Hemenway Falk, and Andreas Haeberlen. Honeycrisp: large-scale differentially private aggregation without a trusted core. In *SOSP*, pages 196–210. ACM, 2019.

[342] Theo Ryffel, Andrew Trask, Morten Dahl, Bobby Wagner, Jason Mancuso, Daniel Rueckert, and Jonathan Passerat-Palmbach. A generic framework for privacy preserving deep learning, 2018.

[343] John K Salmon, Mark A Moraes, Ron O Dror, and David 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*, page 16. ACM, 2011.

[344] Sumudu Samarakoon, Mehdi Bennis, Walid Saad, and Mérouane Debbah. Federated learning for ultra-reliable low-latency V2V communications. *CoRR*, abs/1805.09253, 2018. URL `http://arxiv.org/abs/1805.09253`.

[345] Sai Sri Sathya, Praneeth Vepakomma, Ramesh Raskar, Ranjan Ramachandra, and Santanu Bhattacharya. A review of homomorphic encryption libraries for secure computation. *arXiv preprint arXiv:1812.02428*, 2018.

[346] Felix Sattler, Simon Wiedemann, Klaus-Robert Müller, and Wojciech Samek. Robust and communication-efficient federated learning from non-IID data. *arXiv preprint arXiv:1903.02891*, 2019.

[347] R. Schnell. Efficient private record linkage of very large datasets. In *59^{th} World Statistics Congress*, 2013.

[348] R. Schnell, T. Bachteler, and J. Reiher. A novel error-tolerant anonymous linking code. Technical report, Paper No. WP-GRLC-2011-02, German Record Linkage Center Working Paper Series, 2011.

[349] Claus 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.

[350] SEAL. Microsoft SEAL (release 3.4). https://github.com/Microsoft/SEAL, October 2019. Microsoft Research, Redmond, WA.

[351] Arvind Seshadri, Mark Luk, Adrian Perrig, Leendert van Doom, and Pradeep K. Khosla. Pioneer: Verifying code integrity and enforcing untampered code execution on legacy systems. In *Malware Detection*, volume 27 of *Advances in Information Security*, pages 253–289. Springer, 2007.

[352] Ali Shafahi, Mahyar Najibi, Amin Ghiasi, Zheng Xu, John Dickerson, Christoph Studer, Larry S Davis, Gavin Taylor, and Tom Goldstein. Adversarial training for free. *NeurIPS*, 2019.

[353] Vivek Sharma, Praneeth Vepakomma, Tristan Swedish, Ken Chang, Jayashree Kalpathy-Cramer, and Ramesh Raskar. ExpertMatcher: Automating ML model selection for clients using hidden representations. *arXiv preprint arXiv:1910.03731*, 2019.

[354] Yash Sharma and Pin-Yu Chen. Attacking the Madry defense model with l_1-based adversarial examples. *arXiv preprint arXiv:1710.10733*, 2017.

[355] Yanyao Shen and Sujay Sanghavi. Learning with bad training data via iterative trimmed loss minimization. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 5739–5748, Long Beach, California, USA, 09–15 Jun 2019. PMLR. URL http://proceedings.mlr.press/v97/shen19e.html.

[356] Elaine Shi, HTH Chan, Eleanor Rieffel, Richard Chow, and Dawn Song. Privacy-preserving aggregation of time-series data. In *Annual Network & Distributed System Security Symposium (NDSS)*, 2011.

[357] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 3–18. IEEE, 2017.

[358] Kumar Shridhar, Felix Laumann, and Marcus Liwicki. A comprehensive guide to Bayesian convolutional neural network with variational inference. *arXiv preprint: 1901.02731*, 2019.

[359] Daniel L. Silver, Qiang Yang, and Lianghao Li. Lifelong machine learning systems: Beyond learning algorithms. In *AAAI Spring Symposium Series*, 2013.

[360] Abhishek Singh, Praneeth Vepakomma, Otkrist Gupta, and Ramesh Raskar. Detailed comparison of communication efficiency of split learning and federated learning. *arXiv preprint arXiv:1909.09145*, 2019.

[361] Radu Sion and Bogdan Carbunar. On the computational practicality of private information retrieval. In *Proceedings of the Network and Distributed Systems Security Symposium*, pages 2006–06. Internet Society, 2007.

[362] Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet S. Talwalkar. Federated Multi-Task Learning. In *NIPS*, 2017.

[363] Jake Snell, Kevin Swersky, and Richard S. Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, 2017.

[364] Jasper Snoek, Oren Rippel, Kevin Swersky, Ryan Kiros, Nadathur Satish, Narayanan Sundaram, Mostofa Patwary, Mr Prabhat, and Ryan Adams. Scalable Bayesian optimization using deep neural networks. In *International conference on machine learning*, pages 2171–2180, 2015.

[365] Liwei Song, Reza Shokri, and Prateek Mittal. Privacy risks of securing machine learning models against adversarial examples. In *In Proceedings of the ACM Conference on Computer and Communication Security (CCS)*, 2019.

[366] K Srinathan and C Pandu Rangan. Efficient asynchronous secure multiparty distributed computation. In *International Conference on Cryptology in India*, pages 117–129. Springer, 2000.

[367] Brij Mohan Lal Srivastava, Aurélien Bellet, Marc Tommasi, and Emmanuel Vincent. Privacy-Preserving Adversarial Representation Learning in ASR: Reality or Illusion? In *Annual Conference of the International Speech Communication Association (Interspeech)*, 2019.

[368] Jacob Steinhardt, Pang Wei W Koh, and Percy S Liang. Certified defenses for data poisoning attacks. In *Advances in neural information processing systems*, pages 3517–3529, 2017.

[369] Thomas Steinke and Jonathan Ullman. Tight lower bounds for differentially private selection. In *FOCS*, pages 552–563, 2017.

[370] Sebastian U Stich. Local SGD converges fast and communicates little. In *International Conference on Learning Representations (ICLR)*, 2019.

[371] Sebastian U Stich and Sai Praneeth Karimireddy. The error-feedback framework: Better rates for SGD with delayed gradients and compressed communication. *arXiv:1909.05350*, 2019.

[372] Lili Su and Nitin H. Vaidya. Fault-Tolerant Multi-Agent Optimization: Optimal Iterative Distributed Algorithms. In *PODC*, 2016.

[373] Pramod Subramanyan, Rohit Sinha, Ilia Lebedev, Srinivas Devadas, and Sanjit A Seshia. A formal foundation for secure remote execution of enclaves. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 2435–2450. ACM, 2017.

[374] Ziteng Sun, Peter Kairouz, Ananda Theertha Suresh, and H Brendan McMahan. Can you really backdoor federated learning? *arXiv preprint arXiv:1911.07963*, 2019.

[375] support.google. Your chats stay private while Messages improves suggestions, 2019. URL `https://support.google.com/messages/answer/9327902`. Retrieved Aug 2019.

[376] Ananda Theertha Suresh, Felix X. Yu, Sanjiv Kumar, and H Brendan McMahan. Distributed mean estimation with limited communication. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3329–3337. JMLR. org, 2017.

[377] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *ICLR*, 2013.

[378] Gábor J Székely, Maria L Rizzo, Nail K Bakirov, et al. Measuring and testing dependence by correlation of distances. *The annals of statistics*, 35(6):2769–2794, 2007.

[379] Hanlin Tang, Xiangru Lian, Ming Yan, Ce Zhang, and Ji Liu. D2: Decentralized training over decentralized data. In *ICML*, 2018.

[380] Hanlin Tang, Xiangru Lian, Shuang Qiu, Lei Yuan, Ce Zhang, Tong Zhang, and Ji Liu. DeepSqueeze: Parallel stochastic gradient descent with double-pass error-compensated compression. *arXiv preprint arXiv:1907.07346*, 2019.

[381] Om Thakkar, Galen Andrew, and H Brendan McMahan. Differentially private learning with adaptive clipping. *arXiv preprint arXiv:1905.03871*, 2019.

[382] Florian Tramèr and Dan Boneh. Slalom: Fast, verifiable and private execution of neural networks in trusted hardware. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=rJVorjCcKQ`.

[383] Florian Tramèr and Dan Boneh. Adversarial training and robustness for multiple perturbations. *arXiv preprint arXiv:1904.13000*, 2019.

[384] Florian Tramèr, Fan Zhang, Ari Juels, Michael K. Reiter, and Thomas Ristenpart. Stealing machine learning models via prediction APIs. In *25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016.*, pages 601–618, 2016. URL `https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/tramer`.

[385] Florian Tramèr, Fan Zhang, Huang Lin, Jean-Pierre Hubaux, Ari Juels, and Elaine 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*, pages 19–34, 2017.

[386] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian J. Goodfellow, Dan Boneh, and Patrick 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.

[387] Brandon Tran, Jerry Li, and Aleksander Madry. Spectral signatures in backdoor attacks. In *Advances in Neural Information Processing Systems*, pages 8000–8010, 2018.

[388] Jonathan Ullman. Tight lower bounds for locally differentially private selection. Technical Report abs/1802.02638, arXiv, 2018. URL `http://arxiv.org/abs/1802.02638`.

[389] The Google-Landmark v2 Authors. Google landmark dataset v2, 2019. URL `https://github.com/cvdfoundation/google-landmark`.

[390] Jaideep Vaidya, Hwanjo Yu, and Xiaoqian Jiang. Privacy-preserving SVM classification. *Knowl. Inf. Syst.*, 14 (2), January 2008.

[391] Jo Van Bulck, Marina Minkin, Ofir Weisse, Daniel Genkin, Baris Kasikci, Frank Piessens, Mark Silberstein, Thomas F Wenisch, Yuval Yarom, and Raoul Strackx. Foreshadow: Extracting the keys to the intel {SGX} kingdom with transient out-of-order execution. In *27th {USENIX} Security Symposium ({USENIX} Security 18)*, pages 991–1008, 2018.

[392] Paul Vanhaesebrouck, Aurélien Bellet, and Marc Tommasi. Decentralized collaborative learning of personalized models over networks. In *AISTATS*, 2017.

[393] Praneeth Vepakomma, Otkrist Gupta, Tristan Swedish, and Ramesh Raskar. Split learning for health: Distributed deep learning without sharing raw patient data. *arXiv preprint arXiv:1812.00564*, 2018.

[394] Praneeth Vepakomma, Chetan Tonde, Ahmed Elgammal, et al. Supervised dimensionality reduction via distance correlation maximization. *Electronic Journal of Statistics*, 12(1):960–984, 2018.

[395] Praneeth Vepakomma, Otkrist Gupta, Abhimanyu Dubey, and Ramesh Raskar. Reducing leakage in distributed deep learning for sensitive health data. *arXiv preprint arXiv:1812.00564*, 2019.

[396] Thijs Vogels, Sai Praneeth Karimireddy, and Martin Jaggi. PowerSGD: Practical low-rank gradient compression for distributed optimization. In *NeurIPS 2019 - Advances in Neural Information Processing Systems 32*, 2019.

[397] Riad S. Wahby, Ioanna Tzialla, Abhi Shelat, Justin Thaler, and Michael 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*.

[398] Bolun Wang, Yuanshun Yao, Shawn Shan, Huiying Li, Bimal Viswanath, Haitao Zheng, and Ben Y Zhao. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *2019 IEEE Symposium on Security and Privacy*. IEEE, 2019.

[399] Jianyu Wang and Gauri Joshi. Cooperative SGD: A unified framework for the design and analysis of communication-efficient SGD algorithms. *preprint*, August 2018. URL `https://arxiv.org/abs/1808.07576`.

[400] Jianyu Wang and Gauri Joshi. Adaptive Communication Strategies for Best Error-Runtime Trade-offs in Communication-Efficient Distributed SGD. In *Proceedings of the SysML Conference*, April 2019. URL `https://arxiv.org/abs/1810.08313`.

[401] Jianyu Wang, Anit Sahu, Gauri Joshi, and Soummya Kar. MATCHA: Speeding Up Decentralized SGD via Matching Decomposition Sampling. *preprint*, May 2019. URL `https://arxiv.org/abs/1905.09435`.

[402] Jianyu Wang, Vinayak Tantia, Nicolas Ballas, and Michael Rabbat. SlowMo: Improving communication-efficient distributed SGD with slow momentum. *arXiv preprint arXiv:1910.00643*, 2019.

[403] Kangkang Wang, Rajiv Mathews, Chloé Kiddon, Hubert Eichner, Françoise Beaufays, and Daniel Ramage. Federated evaluation of on-device personalization. *arXiv preprint arXiv:1910.10252*, 2019.

[404] Tongzhou Wang, Jun-Yan Zhu, Antonio Torralba, and Alexei A Efros. Dataset distillation. *arXiv preprint arXiv:1811.10959*, 2018.

[405] Yu-Xiang Wang, Borja Balle, and Shiva Kasiviswanathan. Subsampled R\'enyi differential privacy and analytical moments accountant. *arXiv preprint arXiv:1808.00087*, 2018.

[406] Stanley L. Warner. Randomized response: A survey technique for eliminating evasive answer bias. *Journal of the American Statistical Association*, 60(309):63–69, 1965.

[407] WeBank. WeBank and Swiss re signed cooperation MOU, 2019. URL `https://finance.yahoo.com/news/webank-swiss-signed-cooperation-mou-112300218.html`. Retrieved Aug 2019.

[408] Eric Wong, Frank R Schmidt, and J Zico Kolter. Wasserstein adversarial examples via projected sinkhorn iterations. *ICML*, 2019.

[409] Gavin Wood et al. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, 151(2014):1–32, 2014.

[410] D. Woodruff and S. Yekhanin. A geometric approach to information-theoretic private information retrieval. In *20th Annual IEEE Conference on Computational Complexity (CCC'05)*, pages 275–284, June 2005. doi: 10.1109/CCC.2005.2.

[411] Blake Woodworth, Jialei Wang, H. Brendan McMahan, and Nathan Srebro. Graph oracle models, lower bounds, and gaps for parallel stochastic optimization. In *Advances in Neural Information Processing Systems (NIPS)*, 2018. URL `https://arxiv.org/abs/1805.10222`.

[412] Xiang Wu, Ruiqi Guo, Ananda Theertha Suresh, Sanjiv Kumar, Daniel N Holtmann-Rice, David Simcha, and Felix X. Yu. Multiscale quantization for fast similarity search. In *Advances in Neural Information Processing Systems*, pages 5745–5755, 2017.

[413] Cihang Xie, Yuxin Wu, Laurens van der Maaten, Alan Yuille, and Kaiming He. Feature denoising for improving adversarial robustness. *CVPR*, 2019.

[414] Cong Xie. Zeno++: robust asynchronous SGD with arbitrary number of Byzantine workers. *arXiv preprint arXiv:1903.07020*, 2019.

[415] Cong Xie, Sanmi Koyejo, and Indranil 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.

[416] Cong Xie, Sanmi Koyejo, and Indranil Gupta. Zeno: Distributed stochastic gradient descent with suspicion-based fault-tolerance. In *International Conference on Machine Learning*, pages 6893–6901, 2019.

[417] Sirui Xie, Hehui Zheng, Chunxiao Liu, and Liang Lin. SNAS: stochastic neural architecture search. *arXiv preprint arXiv:1812.09926*, 2018.

[418] Tiancheng Xie, Jiaheng Zhang, Yupeng Zhang, Charalampos Papamanthou, and Dawn Song. Libra: Succinct zero-knowledge proofs with optimal prover computation. In *CRYPTO (3)*, volume 11694 of *Lecture Notes in Computer Science*, pages 733–764. Springer, 2019.

[419] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. *CoRR*, abs/1902.04885, 2019. URL `http://arxiv.org/abs/1902.04885`.

[420] Timothy Yang, Galen Andrew, Hubert Eichner, Haicheng Sun, Wei Li, Nicholas Kong, Daniel Ramage, and Françoise Beaufays. Applied federated learning: Improving Google keyboard query suggestions. *arXiv preprint 1812.02903*, 2018.

[421] Andrew C Yao. Protocols for secure computations. In *Symposium on Foundations of Computer Science*, 1982.

[422] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *FOCS*, pages 162–167. IEEE Computer Society, 1986.

[423] Fangwei Ye, Carolina Naim, and Salim El Rouayheb. Preserving ON-OFF privacy for past and future requests. In *2019 IEEE Information Theory Workshop (ITW)*, August 2019.

[424] Min Ye and Alexander Barg. Optimal schemes for discrete distribution estimation under locally differential privacy. *IEEE Transactions on Information Theory*, 2018.

[425] Samuel Yeom, Irene Giacomelli, Matt Fredrikson, and Somesh Jha. Privacy risk in machine learning: Analyzing the connection to overfitting. In *2018 IEEE 31st Computer Security Foundations Symposium (CSF)*, pages 268–282. IEEE, 2018.

[426] Dong Yin, Yudong Chen, Kannan Ramchandran, and Peter Bartlett. Byzantine-robust distributed learning: Towards optimal statistical rates. In *ICML*, 2019.

[427] Chen Yu, Hanlin Tang, Cedric Renggli, Simon Kassing, Ankit Singla, Dan Alistarh, Ce Zhang, and Ji Liu. Distributed learning over unreliable networks. *arXiv preprint arXiv:1810.07766*, 2018.

[428] Hao Yu, Sen Yang, and Shenghuo Zhu. Parallel restarted SGD for non-convex optimization with faster convergence and less communication. *arXiv preprint arXiv:1807.06629*, 2018.

[429] Hao Yu, Rong Jin, and Sen Yang. On the linear speedup analysis of communication efficient momentum SGD for distributed non-convex optimization. *arXiv preprint arXiv:1905.03817*, 2019.

[430] Muhammad Bila Zafar, Isabel Valera, Manuel Gomez Rodriguez, and Krishna P. Gummadi. Fairness constraints: Mechanisms for fair classification. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, 2017.

[431] Valentina Zantedeschi, Aurélien Bellet, and Marc Tommasi. Fully Decentralized Joint Learning of Personalized Models and Collaboration Graphs. Technical report, arXiv:1901.08460, 2019.

[432] Sixin Zhang, Anna E Choromanska, and Yann LeCun. Deep learning with elastic averaging SGD. In *Advances in Neural Information Processing Systems*, pages 685–693, 2015.

[433] Yu Zhang and Qiang Yang. A survey on multi-task learning. *CoRR*, abs/1707.08114, 2017. URL `http://arxiv.org/abs/1707.08114`.

[434] Yuchen Zhang, John Duchi, Micheal I. Jordan, and Martin J. Wainwright. Information-theoretic lower bounds for distributed statistical estimation with communication constraints. In *Advances in Neural Information Processing Systems*, pages 2328–2336, 2013.

[435] Yawei Zhao, Chen Yu, Peilin Zhao, and Ji Liu. Decentralized online learning: Take benefits from others' data without sharing your own to track global trend. *arXiv preprint arXiv:1901.10593*, 2019.

[436] Michael Zhu and Suyog Gupta. To prune, or not to prune: exploring the efficacy of pruning for model compression. *arXiv preprint arXiv:1710.01878*, 2017.

[437] Wennan Zhu, Peter Kairouz, Haicheng Sun, Brendan McMahan, and Wei Li. Federated heavy hitters discovery with differential privacy. *arXiv preprint arXiv:1902.08534*, 2019.

[438] Xiaojin Zhu. Machine teaching: An inverse problem to machine learning and an approach toward optimal education. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

A 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 [345]:

- TensorFlow Federated [38] specifically targets research use cases, providing large-scale simulation capabilities as well as flexible orchestration for the control of sampling.
- PySyft [342] 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 [35] provides multiple datasets (see below), as well as simulation and evaluation capabilities.

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

- FATE (Federated AI Technology Enabler) [34] is an open-source project intended to provide a secure computing framework to support the federated AI ecosystem.
- PaddleFL [36] is an open source federated learning framework based on PaddlePaddle [37]. In PaddleFL, several federated learning strategies and training strategies are provided with application demonstrations.
- Clara Training Framework [33] includes the support of cross-silo federated learning based on a server-client approach with data privacy protection.

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 [38] 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.[11] At the time of writing, three datasets are supported and we recommend researchers to benchmark on them.

[11] `https://www.tensorflow.org/datasets/api_docs/python/tfds/as_numpy`.

- *EMNIST* dataset [109] 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*[12] 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 [88], 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 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[13] 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 [389] that includes over 5 million images of more than 200 thousand different types of landmark. Similar to 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. [278] has recently published a federated dataset for computer vision. The dataset contains more than 900 annotated street images generated from 26 street cameras and 7 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 [38] 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 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 Section 2.2. For example, a patient may go to one medical institute for a pathology test and go to another for radiology

[12] `https://www.kaggle.com/stackoverflow/stackoverflow`
[13] `https://www.inaturalist.org/`

picture archiving, in which case the features of one sample are partitioned over two institutes regulated by HIPAA. [26].