Federated Learning: What, Why & How?

Abhash Kumar Singh, **Akash Sharma**, **Tarun Bedi**

Abstract

Federated learning is emerging as a new technique enabling distributed clients to collaborate over a shared Machine Learning model while keeping their data localized, by sharing trained weights instead. Implementing and analyzing a federated learning setup still remains a challenge, with multiple parameters and configurations. In this paper, we aim to do a two-part study. First, we aim to do a bird's eye view of the federated learning landscape, and its main challenges. Second, we aim to extend a federated learning simulation system, towards a highly scalable cluster-based environment, to help get a better sense of its advantages in particular machine learning applications.

1. Introduction

The technique of building algorithms that learn from huge volumes of data, and give accurate predictions on unseen data, that is, Machine Learning, has lead to advancements in many application areas. Multitudes of data from varying sources have been used to feed the Machine Learning pipeline, and not only solve the challenging problems but also making applications that provide a great user experience. Sensitive data such as medical-health, one generated from personal devices (IoT devices, mobiles, etc.) is no different, however, there have always been concerns about using it in the correct and optimal way while preserving the user's or organization's privacy.

Federated learning is a Machine Learning technique, that helps train a model over certain decentralized servers, without actually exchanging the data between each other. It leads to learning a global optimized Machine Learning model while avoiding the sharing of respective data samples (keeping them localized) between each of the distributed clients, and helps to learn from non-IID (Independently and Identically distributed) data. Figure 1. shows the basic workflow of Federated learning. The iterative process, where the clients share their respective weights and the server subsequently uses them to train a model, and distributes it back to the clients, continues until the model converges.

Federated learning was first introduced in 2017 and since then, has found its way in multiple high scale applications. One such real-world implementation is the next-word prediction in Google's Android keyboard (Andrew Hard, 2019). Whenever the keyboard shows a recommended result, our personal device stores all the relevant information and context (for e.g. if you clicked the recommended option, etc.) locally. Now, all such device localized history is used for Federated Learning to better the query suggestion model. The overall next word-prediction accuracy increase of 24 percent along with 10 percent more clicks makes Federated learning a strong alternative to the currently used approaches.

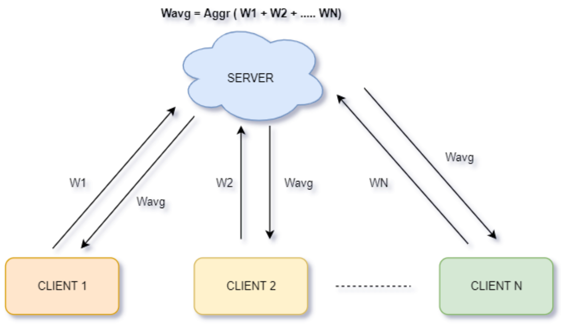

Figure 1. A typical federated learning client-server setup

Federated Learning not only ensures privacy, but also leads to lower latency, more accurate models, and less power consumption. The standard Machine Learning approaches which rely heavily on the common centralized data have large storage requirements. Non-centralized Federated Learning can solve the above problem and help take advantage of the local device resources. Federated learning has led to providing personalized user experiences without compromising user privacy.

Despite the above advantages of Federated Learning over the traditional Machine Learning approaches, there has been little attention on the concepts, advantages, and most importantly the possible applications along with the feasibility in different domains. In this work, we have done a survey of Federated Learning, covering the major work that has been done, and have built on top of a current Federated Learning simulator to assess the impact of various parameters such as latency, number of clients involved, etc. on its performance. We have also covered in brief, the major Federated Learning challenges such as Privacy, Communication efficiency, System Heterogeneity,and Statistical Heterogeneity along with the relevant work that has been done to address them.

2. Survey of Related Work

The idea of Federated Learning was initiated by Google (Jakub Konecný, 2016), where the intention was to learn an optimized global model using localized data on distributed nodes, without actually sharing it with each other. The subsequent work focused on its applications and solving multiple associated challenges, such as security improvisation (Keith Bonawitz, 2017) (Robin C. Geyer, 2017), making it more personal (Fei Chen, 2018), and statistical challenges resolution (Virginia Smith, 2017).

We have summarised below the major challenges associated with Federated Learning and the relevant work done to address them.

2.1. Communication Efficiency

Given that network latencies are the major bottleneck in the federated setup, communication is a key factor to take into account while developing federated learning networks. We discussed some proposed strategies to optimize communication efficiency.

2.1.1. Compression Schemes

While the number of round trips is an important optimization, it is also important to reduce message sizes over the network for reduced latencies. A number of model compression methods have been proposed recently (H. Wang & Wright, 2018) (H. Zhang & Zhang, 2017), and methods such as sparsification and quantization have emerged as top choices.

While these methods are well studied in traditional distributed training, the non-identically distributed data in the federated setup poses interesting challenges. There has been a recent focus on providing stronger compression guarantees for the federated setup (S. Caldas & Talwalkar, 2018) (J. Konecny & Bacon, 2016), and even though there has been theoretical work towards guaranteeing convergence for non-identically distributed data, the works majorly do not take into account practical considerations such as device reliability and locally-updated training methods.

2.1.2. Local Updating

Mini-batch optimization methods have been the popular choice for distributed machine learning but they have shown limited flexibility in terms of distributed data processing (V. Smith & Jaggi, 2018) (Stich, 2019). Instead, approaches which allow multiple local updates to be applied on each device in parallel per communication round, have been shown to achieve orders of magnitude speedups in real world data center environments (S. Zhang & LeCun, 2015) (S. J. Reddi & Smola, 2016). FedAvg, which is based on such local updates, has been the most commonly used method. It has been shown to work well for non-convex problems, but it doesn't provide convergence guarantees.

2.1.3. Decentralized Training

In decentralized training, each device either communicates with a few devices (for example a group of device which are nearby) or with every other device (peer to peer). Decentralized algorithms can be helpful in reducing the communication cost which is required in case there is a central server. Some works have investigated this with linear models(L. He & Jaggi, 2018) while some assume full device participation(A. Lalitha & Koushanfar, 2019). Hierarchical communication patterns having edge servers to compile weights from edge devices and then passing these on to a cloud server, have also been explored, but are not applicable to all networks as this type of hierarchy may not exist or be known beforehand. (T. Lin & Jaggi, 2018) (L. Liu & Letaief, 2019)

2.2. System Heterogeneity

In a federated learning network, different devices are involved in learning an optimized global model. These devices may differ in their communication potential, computational capacity, and storage power. These differences in underlying machines are mainly caused due to different system specifications such as CPU, inherent memory, connectivity ranges (4G, 5G), etc (Tian Li, 2019). Also, some erratic devices may drop out of the system, or may not participate (K. Bonawitz, 2019) and might eventually impact the overall accuracy of the global model obtained.

The capability to deal with such erratic and unpredictable situations arising in a distributed environment is in itself a challenge for Federated Learning and some relevant work for each major challenge has been discussed below.

2.2.1. Low Participation

In Federated Learning, it might be possible to have only a handful of nodes participating at the same time, due to the size of network, and the system-level issues. Some of these challenges have been talked about in (Tian Li, 2019) (Peter Kairouz, 2019). Other relevant paper that has tried to address the problem is (Tao Yu, 2020), and has proposed approaches by which all the individual clients can benefit using local adaptation. Some of the studies have addressed dynamic scheduling as the solution and recommend proactive adaptation as the participation changes (Qinbin Li, 2020). This will help the network adapt at run-time, and handle the inaccuracy surge that comes as a result.

2.2.2. Heterogeneous Hardware

The underlying devices in a federated network may differ in their hardware specifications, wireless connectivity ranges, and CPU capabilities (Tian Li, 2019). Some studies carried out experiments, on the efficient utilization of limited node resources (Shiqiang Wang, 2019), to achieve a similar performance compared to other clients involved. Also, to cater to the need, some work has focused on reducing the client resource requirements (Sebastian Caldas, 2018). This is still an open problem in active research.

2.2.3. Dropped Network Devices

The device dropouts during model training is a very common problem faced in existing Federated Learning systems. This might result due to connectivity issues or energy constraints (Tian Li, 2019). Making federated Learning more robust, stable, and fault-tolerant to such a challenge has been summarised by many major works, such as (Avishek Ghosh, 2019). The noise generation mechanism must be able to handle this either by increasing the per client noise (Peter Kairouz, 2019) or having a node connection protocol over several times in the training. (Keith Bonawitz, 2019), that is the Tensorflow Federated Learning model automatically takes care of such dropped devices in the network and is a good step towards robust Federated Learning.

2.3. Statistical Heterogeneity

The most fundamental assumption we take before approaching a machine learning problem is that the data is independent and identically distributed(IID). Problems come up when training federated models from data which is not IID across devices, in terms of analyzing convergence and modeling the data.

2.3.1. Convergence for non-IID data

Methods like FedAvg (H. B. McMahan & y Arcas, 2017) are shown to diverge in case of non-IID data. Another method, FedProx (T. Li & Smith, 2018), accounts for systems heterogeneity across devices by making changes in FedAvg to ensure convergence. While some works (Jiang & Agrawal, 2018) (S. Wang & Chan, 2019) have explored convergence with assumptions like convexity or uniformly bounded gradients, some have pursued heuristic approaches (L. Huang & Liu, 2018) (Y. Zhao & Chandra, 2018) like sharing local device data or server side data. However, these approaches violate the privacy assumption of federated learning while adding to network bandwidth and need to be carefully managed.

2.3.2. Modeling disparate data

Simply solving a loss function like below :

$$\min_w F(w),\ F(w) = \sum_{k=1}^{m} p_k \cdot F_k(w),$$

where, $p_k \geq 1, \sum_k p_k = 1$, m is the total number of devices, F_k is the local objective function for kth device, may advantage/disadvantage some devices. This can happen when the model becomes biased towards devices with more data, or if devices are weighed equally, it can be biased towards groups of devices which occur more. Recent works have been done to reduce the variance of the model performance across devices. (L. Huang & Liu, 2018) performs local updates based on local loss. Agnostic federated learning (M. Mohri & Suresh, 2019) uses a minimax optimization scheme for a data distribution formed by a combination of client data distributions. (T. Li & Smith, 2018) suggests a method in which devices with higher loss are given higher weight to have less variance in the final accuracy distribution. Methods such as meta-learning and multi-task learning have been extended to federated setting. (V. Chen & Raykova, 2019) models star topology as a Bayesian network and can handle non-convex objective functions, although being expensive to generalize to large networks. (H. Eichner & Talwar, 2019) propose a solution which adaptively chooses between a global and device-specific models to deal with cyclic patterns in data samples. MOCHA (V. Smith & Talwalkar, 2017), an optimization framework for federated learning, allows learning separate but related models for each device. It makes use of multi-task learning and provides convergence guarantees for only convex objectives, but is not scalable to large networks. Despite the recent research, approaches to model disparate data that is robust and scalable pose challenges that need to be solved.

2.4. Privacy

While privacy concerns are the usual motivation to keep data on local devices and adopt the federated setup, a lot of recent work demonstrates how sharing other information like model weight updates could potentially leak confidential user information. (Abhishek Bhowmick, 2019) (Nicholas Carlini, 2019) (Matt Fredrikson, 2015) (Arjun Nitin Bhagoji, 2018) (Eugene Bagdasaryan, 2019)

Below, we first summarize the major privacy concerns, followed by recent privacy-preserving methods developed specially for the federated learning setup.

2.4.1. Privacy concerns

- **Interception of user data summaries:**
  When sending over a set of weights from a client to the server, the very first concern is of network interception - where this data is instead accessed by an attacker. To solve for this problem, the transmitted data is usually encrypted using cryptographic key pairs.

- **Confidential information leak:**
  As highlighted by many recent works, notably in (Nicholas Carlini, 2019), a recent cause of concern is a potential leak of sensitive user information when the client sends a set of weights to the server. While this area is still in active research, the commonly used solution is of differential privacy, where clients add a certain noise to the weights being set out from the client to the server.

- **Model poisoning:**
  Another recent work (Eugene Bagdasaryan, 2019) highlights the concern of model poisoning and backdoors, where malicious clients could potentially carefully-craft data points and send over these weights to adversarially fool the global model, or create backdoors which would lead to misclassification. The major reason for this behavior is that the server has no control over the client behavior and their data distributions, and building federated systems to be more robust to such misbehaving clients is still in active research.

2.4.2. Differential privacy

Among the various privacy approaches studied, differential privacy is the most widely used due to its small system overheads and strong theoretical guarantees. (C. Dwork, 2014) (C. Dwork, 2006)

On a high level, a training mechanism is said to be differentially private if a small change in the input distribution does not result in too much of change in the output distribution, making it harder to gain insights of the actual sensitive training data used if given a set of trained weights.

Essentially, differential privacy provides a guarantee that no or little information about an individual data point is revealed while sharing training results from a client to the server. For gradient-based machine learning methods, a common way to apply differential privacy is by perturbing the intermediate output at each iteration (M. Abadi, 2016), e.g. via Gaussian, Binomial or Laplacian noise.

3. Problem Statement

Even with a lot of recent work highlighting challenges in the federated setup, it still largely remains unclear as to how the setup will behave in particular machine learning applications - especially with a number of parameters, including individual client failures and network latencies, out of the systems' control. With a lot of missing high-level clarity, we feel that using a simulation system with all these practical factors as configurational parameters would really help to get a sense of the system's potential behavior.

We believe that this seemingly simple simulation system would be beneficial to evaluate the feasibility of implementing a federated learning system in a number of variable circumstances, and would help gain more clarity of the systems behavior.

4. Setup

We have built on top of the proposed PrivacyFL simulator system (Vaikkunth Mugunthan, 2020) for running our experiments. It is a flexible and highly configurable simulator for Federated Learning environments, with key features including latency simulation, support for both centralized (with central servers) and decentralized (serverless) learning and configurable security and privacy mechanisms. The working of the simulator is described below.

The simulator provides a simple and lightweight Python framework to simulate privacy-preserving secure Federated Learning.

- A client corresponds to an instance of the ClientAgent class. Each client is capable of interacting with other clients.
- Client agents can also talk to ServerAgent class if the setup is a centralized one.
- The server agent doesn't train models but serves to run the federated learning algorithm. It requests weight from clients, averages them, and returns the federated weights to clients.
- To start the simulation, first, an instance of Initializer class is created. This creates the agents in the simulation and invoke client agents to perform other tasks like key exchanges. The initializer gives each agent a data structure containing a mapping of agent names and instances, enabling communication between them. Also, it gives each client a ModelEvaluator instance for test dataset evaluation.

4.1. Classes

Two main classes involved are ClientAgent and ServerAgent.

- **ClientAgent Class:** This is a subclass of **Agent** class. It has mainly three methods :

- **produce_weights(self, message):** It is called every iteration and tells the client to train its machine learning model on its dataset for that iteration. The message contains information like iteration number, simulated time, etc.
- **receive_weights(self, message):** When the server has federated weights to return to the client, this method is called. The message has information about iteration and weights. This method returns True if the weights of the client have converged with the federated weights and False otherwise.
- **remove_active_clients(self, message):** It is used when simulation is configured with client drop out. This is called at the end of the iteration. The message contains the information about iteration and clients that have dropped out.

- **ServerAgent Class:** This class looks after running the federated learning algorithm. It has one method:
  - **request_values(num_iterations):** Initializer calls this method to begin requesting clients' weights. request_values() first requests weights from the clients and calls their produce_weights(). Then it averages the weights received and returns the federated weights to the clients by calling their receive_weights() method. If the configuration allows client dropout, the server calls the clients' remove_active_clients() method.

4.2. Simulation Workflow

- First, changes are made in config.py as per usage and run_simulation.py is run.
- run_simulation.py creates Initializer, which in turn creates the client agents and server agent. It calls the initializer's run_simulation(), which next calls the server agent's request_values() method.
- For each iteration,
  - Server calls each client's produce_weights().
  - Each client then trains its machine learning model on the dataset for that iteration, saves the weights locally, and then creates a copy of its weights added with security offset and differential private noise. These modified weights are sent to the server.
  - The server receives these weights, averages them and returns federated weights to each client through receive_weights().
  - Each client then calculates test accuracy by using it local and federated weights. If it matches the convergence criteria, it notifies the server whether it's dropping out from further training.

5. Experiments

For our evaluation, we trained the clients on UCI's Skin Segmentation dataset(Rajen Bhatt, 2012) for 10 iterations. All experiments were run locally on a MacBook Pro 1.4GHz Intel Core i5 with 16GB RAM.

- **Iteration-wise accuracies:**
  With a 3-client 1-server model, we try to study the iteration-wise accuracy-trends of each client and the server over the federated training cycle.
  As in figure 2, we observe that maintaining a federated model helps all clients, especially initially poorly performing ones, approach to the common federated accuracy.

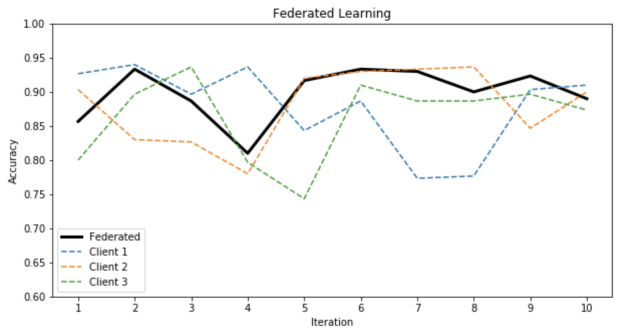

Figure 2. Federated learning - 3 clients / 1 server evaluation on a Skin Segmentation dataset (Rajen Bhatt, 2012)

- **Decentralized Federated learning:**
  With a 3-client model, we now try to study the iteration-wise accuracy-trends of each client in a decentralized setup. With no server present, each client now sends its trained weights to all other clients, which do an averaging on their end.
  As in figure 3, we observe a similar trend of the global federated weights helping all clients, however, it seems like it takes more iterations for clients to reach to a common federated accuracy.

- **Simulated Latencies:**
  Given that network latencies are the biggest bottleneck in a federated setup, we aim to study the client-training times and the time taken to receive federated weights - when we add in simulated latencies via a simulated latency map, as shown in table 3. We observe in the below tables 1 and 2, that network latencies dominate in time over the training times - given that our ML application is not computationally expensive.

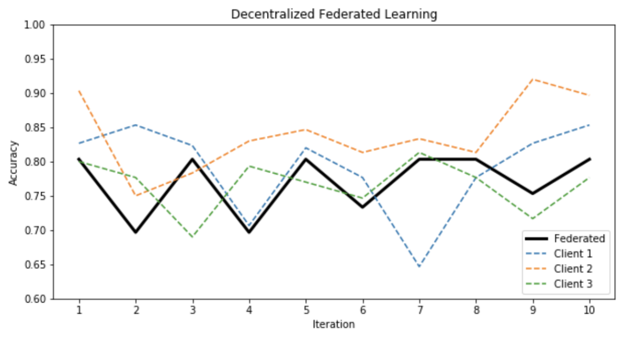

Figure 3. Decentralized federated learning - 3 clients / decentralized evaluation on a Skin Segmentation dataset (Rajen Bhatt, 2012)

Table 1. Simulated time to receive federated weights (in seconds)

Iteration	Client 1	Client 2	Client 3
1	4.30	6.01	4.10
2	4.31	6.00	4.11
3	4.30	6.01	4.11
4	4.31	6.01	4.11

6. Future Work

Federated learning is still in active research, with multiple directions of open challenges. After building over a recently proposed federated learning simulation system, we feel that we now have a strong foundation to meaningfully contribute to some open problems, that we describe below:

- **Simulations at scale:**
  We plan to expand on our simplistic client-server model from our local environment, to cluster-environments, to enable similar simulations at a much higher scale. In addition, we plan to benchmark on more compute-intensive tasks with deeper models, and we plan to place clients and servers in cloud machines at different network locations, so as to have a better sense of actual network latencies.

- **Adversarial machine learning:**
  With the server having no control over the data distribution the client is training over, there has been a lot of recent work highlighting adversarial challenges with the federated setup(Arjun Nitin Bhagoji, 2018) (Eugene Bagdasaryan, 2019), where a malicious client could potentially fool models or create backdoors. As a part of our system, we aim to explore the idea of a smarter weight averaging on the server - where we could potentially detect and skip over some client weights that look like anomalies and could be potentially adversarial.

Table 2. Personal computation time to produce weights (in seconds)

Iteration	Client 1	Client 2	Client 3
1	0.002	0.003	0.002
2	0.008	0.005	0.005
3	0.004	0.003	0.006
4	0.002	0.003	0.006

Table 3. Simulated Delay (in seconds)

	Client 1	Client 2	Client 3
SERVER	0.1	2.0	0.3

- **Hybrid topologies:**
  Given that network latencies are the biggest bottleneck in the federated setup, one largely unexplored idea is of hybrid topologies, where we try to have some parts of the network being decentralized and other parts as centralized. This might be particularly beneficial in cases where the server is far from clusters of clients close to each other and can help save over expensive network requests by instead talking to closeby clients. We feel that this use-case is pretty close to actual implementations at scale and hence aims to explore it as a part of our simulation system.

7. Conclusion

In this paper, we explored the recent ideas of the federated learning setup and looked at some high-level challenges, notably within networking and privacy. In addition, we built over a proposed federated learning simulation system, to study the actual behavior of such a system in a real-world application as well as the effect of network latencies. We aim to make our contributions more relevant to the community, by continuing with our future work, and particularly doing these simulations on a much larger scale.

Acknowledgement

The authors would like to thank Professor Theodoros Rekatsinas for his tremendous patience, experience, guidance, and valuable inputs during the project.

Team Contributions

Our major tasks involved the survey and analysis of Federated Learning, design and setup of the simulation, and carrying out the experiments assessing several parameter variations on Federated learning accuracy and training time. Most of the work related to the survey of Federated learning, its analysis, and simulation setup-design was a collaborative effort. The individual experiments involving the assessment of Federated learning with a centralized server was heavily worked on by Abhash, experiments assessing Federated learning without a central server (decentralized) were heavily worked on by Akash, and the experiments related to latency simulation was heavily worked on by Tarun. However, all of us collaborated on each part so that we learn the project end to end, and shared ideas to make it better.

References

A. Lalitha, X. Wang, O. K. Y. L. T. J. and Koushanfar, F. Decentralized bayesian learning over graphs, arxiv preprint arxiv:1905.1046, 2019.

Abhishek Bhowmick, John Duchi, J. F. G. K. R. R. Protection Against Reconstruction and Its Applications in Private Federated Learning, 2019.

Andrew Hard, Kanishka Rao, R. M. S. R. F. B. S. A. H. E. C. K. D. R. Federated learning for mobile keyboard prediction, 2019.

Arjun Nitin Bhagoji, Supriyo Chakraborty, P. M. S. C. Analyzing Federated Learning through an Adversarial Lens, 2018.

Avishek Ghosh, Justin Hong, D. Y. K. R. Robust federated learning in a heterogeneous environment, 2019.

C. Dwork, A. R. The algorithmic foundations of differential privacy. Foundations and Trends in Theoretical Computer Science, 2014.

C. Dwork, F. McSherry, K. N. A. S. Calibrating noise to sensitivity in private data analysis, 2006.

Eugene Bagdasaryan, Andreas Veit, Y. H. D. E. V. S. How To Backdoor Federated Learning, 2019.

Fei Chen, Zhenhua Dong, Z. L. X. H. Federated meta-learning for recommendation. corr abs/1802.07876 (2018). arxiv:1802.07876, 2018.

H. B. McMahan, E. Moore, D. R. S. H. and y Arcas, B. A. Communication-efficient learning of deep networks from decentralized data. in conference on artificial intelligence and statistics, 2017.

H. Eichner, T. Koren, H. B. M. N. S. and Talwar, K. Semi-cyclic stochastic gradient descent. in international conference on machine learning, 2019.

H. Wang, S. Sievert, S. L. Z. C. D. P. and Wright, S. Atomo: Communication-efficient learning via atomic sparsification, 2018.

H. Zhang, J. Li, K. K. D. A. J. L. and Zhang, C. Zipml: Training linear models with end-to-end low precision, and a little bit of deep learning, 2017.

J. Konecny, H. B. McMahan, F. X. Y. P. R. A. T. S. and Bacon, D. Federated learning: strategies for ‘ improving communication efficiency, 2016.

Jakub Konecný, H. Brendan McMahan, D. R. P. R. Federated optimization: Distributed machine learning for on-device intelligence. corr abs/1610.02527 (2016). arxiv:1610.02527 http://arxiv.org/abs/1610. 02527, 2016.

Jiang, P. and Agrawal, G. A linear speedup analysis of distributed deep learning with sparse and quantized communication. in advances in neural information processing systems, 2018.

K. Bonawitz, H. Eichner, W. G. D. H. A. I. V. I. C. K. J. K. S. M. H. B. M. T. V. O. D. P. D. R. J. R. Towards federated learning at scale: system design, 2019.

Keith Bonawitz, Vladimir Ivanov, B. K. A. M. H. B. M. S. P. D. R. A. S. K. S. Practical secure aggregation for privacy-preserving machine learning, 2017.

Keith Bonawitz, Hubert Eichner, W. G. D. H. A. I. V. I. C. K. J. K. S. M. H. B. M. Towards federated learning at scale: System design, 2019.

L. He, A. B. and Jaggi, M. Cola: Decentralized linear learning. in advances in neural information processing systems, 2018.

L. Huang, Y. Yin, Z. F. S. Z. H. D. and Liu, D. Loadaboost: Loss-based adaboost federated machine learning on medical data. arxiv preprint arxiv:1811.12629, 2018.

L. Liu, J. Zhang, S. S. and Letaief, K. B. Edge-assisted hierarchical federated learning with non-iid data. arxiv preprint arxiv:1905.06641, 2019.

M. Abadi, A. Chu, I. G. H. B. M. I. M. K. T. L. Z. Deep learning with differential privacy, 2016.

M. Mohri, G. S. and Suresh, A. T. Agnostic federated learning. in international conference on machine learning, 2019.

Matt Fredrikson, Somesh Jha, T. R. Model Inversion Attacks that Exploit Confidence Information and Basic Countermeasures, 2015.

Nicholas Carlini, Chang Liu, E. J. K. D. S. The Secret Sharer: Evaluating and Testing Unintended Memorization in Neural Networks, 2019.

Peter Kairouz, H. Brendan McMahan, B. A. A. B. M. B. A. N. B. K. B. Z. C. G. C. R. C. R. G. D. S. E. R. D. E. J. G. Z. G. A. G. B. G. P. B. G. M. G. Z. H. C. H. L. H. Z. H. B. H. J. H. M. J. T. J. G. J. M. K. J. K. A. K. F. K. S. K. T. L. Y. L. P. M. M. M. R. N. A. R. P. M. R. H. Q. D. R. R. R. D. S. W. S. S. U. S. Z. S. A. T. S. F. T. P. V. J. W. L. X. Z. X. Q. Y. F. X. Y. H. Y. S. Z. Advances and open problems in federated learning, 2019.

Qinbin Li, Zeyi Wen, Z. W. S. H. N. W. B. H. A survey on federated learning systems: Vision, hype and reality for data privacy and protection, 2020.

Rajen Bhatt, A. D. Skin Segmentation Dataset, UCI Machine Learning Repository, 2012.

Robin C. Geyer, Tassilo Klein, M. N. Differentially private federated learning: A client level perspective. corr abs/1712.07557 (2017). arxiv:1712.07557, 2017.

S. Caldas, J. Konecny, H. B. M. and Talwalkar, A. Expanding the reach of federated learning by reducing client resource requirements, 2018.

S. J. Reddi, J. Konecny, P. R. B. P. and Smola, A. Aide: Fast and communication efficient distributed ‘ optimization. arxiv preprint arxiv:1608.06879, 2016.

S. Wang, T. Tuor, T. S. K. K. L. C. M. T. H. and Chan, K. Adaptive federated learning in resource constrained edge computing systems. journal on selected areas in communications, 37:1205–1221, 2019.

S. Zhang, A. E. C. and LeCun, Y. Deep learning with elastic averaging sgd. in advances in neural information processing systems, 2015.

Sebastian Caldas, Jakub Konečny, H. B. M. A. T. Expanding the reach of federated learning by reducing client resource requirements, 2018.

Shiqiang Wang, Tiffany Tuor, T. S. K. K. L. C. M. T. H. K. C. Adaptive federated learning in resource-constrained edge computing systems. ieee journal on selected areas in communications, 2019.

Stich, S. U. Local sgd converges fast and communicates little. in international conference on learning representations, 2019.

T. Li, A. K. Sahu, M. S. M. Z. A. T. and Smith, V. Federated optimization for heterogeneous networks arxiv preprint arxiv:1812.06127, 2018.

T. Lin, S. U. S. and Jaggi, M. Don’t use large mini-batches, use local sgd. arxiv preprint arxiv:1808.07217, 2018.

Tao Yu, Eugene Bagdasaryan, V. S. Salvaging federated learning by local adaptation, 2020.

Tian Li, Anit Kumar Sahu, A. T. V. S. Federated learning:challenges, methods, and future directions, 2019.

V. Chen, V. P. and Raykova, M. Secure computation for machine learning with spdz. arxiv preprint arxiv:1901.00329, 2019.

V. Smith, S. Forte, C. M. M. T. M. I. J. and Jaggi, M. Cocoa: a general framework for communicationefficient distributed optimization. journal of machine learning research, 18:1–47, 2018.

V. Smith, C.-K. Chiang, M. S. and Talwalkar, A. Federated multi-task learning. in advances in neural information processing systems, 2017.

Vaikkunth Mugunthan, Anton Peraire-Bueno, L. K. Privacyfl: A simulator for privacy-preserving and secure federated learning, 2020.

Virginia Smith, Chao-Kai Chiang, M. S. A. S. T. Federated multi-task learning. in advances in neural information processing systems 30, i. guyon, u. v. luxburg, s. bengio, h. wallach, r. fergus, s. vishwanathan, and r. garnett (eds.). curran associates, inc., 4424–4434. http://papers.nips.cc/paper/, 2017.

Y. Zhao, M. Li, L. L. N. S. D. C. and Chandra, V. Federated learning with non-iid data. arxiv preprint arxiv:1806.00582, 2018.