CIS 700 Spring 2020
Asynchronous Federated Learning - Literature Review

David Castello
College of Engineering and Computer Science
Syracuse University
drcastel@syr.edu

Abstract

Federated learning is an emergent field of research that seeks to train performant machine learning models across a heterogeneous collection of mobile devices, each with access to private data. Despite the fact that such devices have uneven hardware capabilities and quantities of training data, and that network constraints and hardware failures can occur at any time, state of the art algorithms to conduct federated learning operate in synchronous rounds of communication. This work is a literature review of recent efforts to address this contradiction via novel protocols that relax the need to synchronize local training. Following a background study of the influences behind federated learning, six new research papers are surveyed to assess the progress of this new field towards developing robust and effective asynchronous federated learning algorithms.

Introduction

Modern mobile devices have unprecedented access to boundless amounts of data that, while invaluable to machine learning, cannot be privately transferred or aggregated. Meanwhile, these devices enjoy burgeoning computational power as highly efficiency systems-on-chip become more capable with each passing year. Federated learning is an emergent field of research that seeks to capitalize on the intersection of these two paradigms, pioneering a derived form of distributed optimization that factors in the unique challenges and opportunities of the setting.

In the years following its introduction, FederatedAveraging has remained the baseline algorithm in the federated learning community. It operates synchronously, at each round choosing a subset of client devices to perform training on local data and report back updates to be aggregated into a global model. This mode of operation, however, would appear to be at odds with the setting in which FederatedAveraging was designed to operate. Clients in federated learning are assumed to be a heterogeneous collection of mobile devices with uneven size and distribution of local data, and unequal access to battery power, network bandwidth, and compute power. Further, such devices must be expected to enter or leave a ready state at any time, whether that be because a user makes a demand on their local resources or they experience a hardware failure. Even in light of these realities, FederatedAveraging proceeds in synchronous rounds, forcing the whole learning process to stall while waiting for the selected subset of client devices to either respond or time out. Not only does this limit the number of devices that can be selected at any one time, if too many devices fail to respond an entire round of learning can be lost.

Asynchronous federated learning is an entirely new field of research, increasingly active within the last year, that addresses this contradiction by relaxing the synchronization constraint on new algorithms to perform federated learning. This work is a literature review of this exciting new topic, and the remainder is structured as follows: a brief review of federated learning and the FederatedAveraging algorithm, a deeper dive into the background influences of federated learning and what lessons they reveal about asynchronous designs, a further look at the advantages asynchronicity can provide in federated learning, and finally a discussion of six selected works, the novel protocols they describe, unique capabilities they introduce, and their results.

Review of Federated Learning

Prior to the introduction of federated learning, H. Brendan McMahan and Daniel Ramage at Google, joined by Jakub Konečný from the University of Edinurgh, establish federated optimization as a branch of distributed optimization in (Konečný, McMahan, and Ramage 2015). They describe a setting in which privacy-sensitive information resides on users' mobile devices, which together form a loose "federation" of clients capable of performing a distributed optimization task on their local private data. While existing work had already begun to explore learning in the presence of communication constraints, federated optimization imposes additional criteria that are distinct from traditional distributed optimization. Training data in federated optimization is assumed to be non-IID, unbalanced (some clients will have more training data than others), and massively distributed (likely to be more clients than there are training examples per client). This early research succeeded in identifying a uniquely challenging and rewarding domain, but federated optimization was a novel formulation in need of a novel algorithm.

Joined by others at Google, McMahan and Ramage formalize federated learning a year later in (Brendan McMahan et al. 2016). In addition to reiterating the setting of federated optimization, Brendan McMahan et al. describe a synchronous protocol and algorithm for federated learning that remains the industry standard to this day. The FederatedAveraging algorithm consists of synchronous rounds of communication; in each round, the server selects a random subset of client devices and sends those clients the current global model parameters. The client devices then perform multiple passes of training on their local datasets and send their updated parameters back to the server to be averaged. The server averages the parameters received from the current round of clients to obtain a new state of the global model, beginning the next round of training. The structure of this protocol, in particular the decision to implement it as synchronous schema, is best understood in light of the preceding research that laid the foundation.

Background

Exploring the origins and inspirations of federated learning helps to contextualize the design decisions made by the authors at Google, and serves as a backdrop to the review of asynchronous techniques which refine or change some of these decisions. Federated learning has roots in the optimization community, which seeks to learn low-error hypotheses over data that can't be aggregated due to storage or communication costs, the data center community, primarily concerned with leveraging massively-parallel hardware to achieve learning speedups or to train huge models with millions of parameters, and the privacy community, where the challenging is learning without the leakage of sensitive information owned by individuals or discrete entities. Both asynchronous and synchronous protocols have been proposed across all three settings, where they exhibit many of the same trade-offs that can be seen in federated learning.

Distributed Optimization

Balcan et al.'s contribution to distributed optimization predates the near universal accessibility of powerful mobile hardware that inspired federated learning. Instead, the authors are mainly concerned with the database setting, citing cases where genomic data may be distributed across research teams around the world, or different retailers may maintain separate data on their respective clientele. Importantly, Balcan et al. discuss the case of hospital datasets, where each hospital may have data from a different distribution and privacy concerns are paramount. Both factors are fundamental to federated learning. Moreover, the techniques discussed in this paper are largely asynchronous and bear some resemblance to modern, more sophisticated algorithms.

The baseline approach is for the "center" (server, in the federated setting) to send its most recent "hypothesis" (global model) to each "player" (client), which will return any counterexamples from its local dataset. Direct sharing of private local data was problematic even at the time, and the authors propose refined protocols that circumvent that disclosure and provide lower bounds for communication costs in various settings. Most relevant to this review are those protocols that highlight the trade-offs between synchronous and asynchronous designs. In a synchronous model, the lack of a message sent during a specific round of communication is itself a message. A round-robin algorithm that terminates when no players have a specific local example, for instance, may require additional communication rounds to implement asynchronously. This communication efficiency comes at the cost of increased design and implementation complexity when compared to simpler and more flexible asynchronous standards.

Data Centers

Contemporary to the previous work, the data center community was also exploring distributed machine learning with the aim of increased training performance on very large models. Dean et al. propose "DistBelief", a framework that supports not only parallelism of a single model within and across machines, but also the duplication of models into multiple running instances that receive distributed portions of the learning task. While all model instances are co-located and run on trusted hardware, the structure of the framework bears more than a passing resemblance to federated learning. The similarities continue as the authors describe "Downpour SDG", an asynchronous form of stochastic gradient descent designed to run across DistBelief model instances. Downpour SDG portions the training dataset to multiple instances, which communicate updates asynchronously via a shared parameter server. The parameter server is itself asynchronous, distributed across machines as "shards". These two sources of asynchronicity, model instances and parameter shards, prove to be an asset to the learning process in expected and unexpected ways.

Asynchronicity in DistBelief provides robustness to differences in communication speed and even hardware failure, factors which threaten to bottleneck or halt synchronous designs. Asynchronous processing also introduces additional stochasticity into the learning process via what might traditionally be considered consistency errors: out of sequence or outdated values between models and parameter servers. The benefits of both robustness and added stochasticity are reflected in future work. Although Brendan McMahan et al. ultimately select a state-of-the-art synchronous form of data center stochastic gradient descent to use as their baseline, the research community has been active on this topic in the years since. Recent work such as (Zheng et al. 2016), updated in February of this year, argue that asynchronous approaches remain competitive in data centers. This suggests that advantages found in DistBelief and Downpour SDG may yet apply to federated learning.

Privacy

Distribution in the privacy setting shares many features with work done by the optimization and data center communities, though the end goals are different. (Pathak, Rane, and Raj 2010) is an early work that begins to tackle the root motivations of federated learning through the lens of privacy. The objective is to learn a classifier over a multi-party collection of private data, though unlike in federated learning, the parties involved are assumed to be a relatively small number of data aggregators and the data itself is IID. The authors describe a protocol in which each party learns an optimal classifier on their local data, then securely shares that classifier to an untrusted mediator. The mediator then aggregates the local classifiers using a single-shot noisy averaging procedure to produce a differentially-private classifier over the whole dataset. Averaging in one shot circumvents any need for synchronization, but later research finds that it to be frequently constrained by the worst performing of the local classifiers.

Inspired by the data center parallelization of (Balcan et al. 2012), Shokri and Shmatikov author a seminal work that shares much in common with federated learning. The setting is similar to prior work, noting the privacy risks of centralized storage of sensitive user information and proposing a framework to allow for distributed learning across data stored locally. The key innovation is the asynchronous sharing of a selection of parameters with a central parameter server, facilitating a distributed form of stochastic gradient descent ("Selective Stochastic Gradient Descent", or SSDG) that avoids the direct data transmission (or cryptographic costs) of (Balcan et al. 2012), the communication overhead of (Dean et al. 2012), and the problematic one-shot averaging of (Pathak, Rane, and Raj 2010).

Like in Downpour SDG, here too the asynchronous operation of the parameter server is found to be beneficial to the learning process, where race conditions, network errors, and hardware failures all serve as added stochasticity. This stochasticity, the authors note, bears resemblance to standard regularization techniques that randomly corrupt neurons or input data to prevent overfitting. This theory is put this to the test in experimental evaluation, where Shokri and Shmatikov compare asynchronous parameter sharing with a round-robin synchronous implementation. They find that round-robin sharing produces slightly more accurate models, but at a speed dictated by the slowest of the participants. The authors suggest that synchronous protocols are best suited to scenarios where all participants have a similar hardware capacity, and go on to state that, "the promising accuracy of this protocol indicates that [distributed] SSGD should work well even with unreliable (e.g., mobile) networks" (Shokri and Shmatikov 2015). Given the conceptual similarities between this work and federated learning, it follows that the synchronization trade-offs observed here would apply to the federated setting, which is uniquely characterized by unreliable mobile networks.

Literature Review

Owing to the nascency of the field, much of the current work on asynchronous federated learning is written as if in a vacuum. These papers, all of them published within the past year, represent early forays into a concept that has yet to become a conversation, largely absent the benefit of responding to and building upon earlier related works. Indeed, only one paper in the body of work studied makes reference to another, leading to a substantial amount of repetition as the research community begins to form a foundation on which to build. Given this structure, the literature review will follow a parallel approach, discussing those elements in common to each work and highlighting standout differences. For convenience, Table 1 lists each paper, the authors, and the name of the protocol being proposed.

Title	Authors	Protocol
Asynchronous Federated Optimization	Xie, Koyejo, and Gupta	FedAsync
Asynchronous Federated Learning for Geospatial Applications	Sprague et al.	Async
Asynchronous Online Federated Learning for Edge Devices	Chen, Ning, and Rangwala	ASO-fed
Asynchronous Federated Learning with Differential Privacy for Edge Intelligence	Li et al.	MAPA
Semi-Asynchronous Protocol for Fast Federated Learning with Low Overhead	Wu et al.	SAFA
Communication-Efficient Federated Deep Learning with Asynchronous Model Update and Temporally Weighted Aggregation	Chen, Sun, and Jin	TAWFL

Table 1: Papers covered in this literature review.

Motivations for Asynchronous Operation

From the background research that set the stage for federated learning, certain trade-offs between synchronous and asynchronous designs can be expected. Concern for communication overhead among distributed optimization research, hardware failure in the data center community, and heterogeneity of participating learners as seen in privacy research are all reflected in studies of asynchronous federated learning, in addition to new, unique considerations.

(Xie, Koyejo, and Gupta 2019) notes that in a synchronous protocol, the need for all selected clients to communicate their updated local models at once leads to network congestion that limits the number of clients that can participate each round. Meanwhile, (Wu et al. 2019) observes that the server sending the latest global model to each client in a round presents similar challenges. For those clients actively training their local model at a given time, each paper in this review cites the so-called "straggler effect" that emerges from unrealistic assumptions in FederatedAveraging.

Given the setting that inspired federated learning - building a global model across a loose federation of devices each with access to local data - the authors studied in this review would assert that it is unreasonable to assume equal compute power across edge devices. Compounded with communication slowdowns, network congestion, hardware failure, battery capacity, and other such factors, maintaining federated learning at scale and speed in a synchronous fashion begins to seem like an onerous proposition. Waiting for slower clients (stragglers) to finish training in a given round affects the "wall-clock" efficiency of FederatedAveraging, and dropping those clients that fail to report in time or experience failure mid-training affects the resulting global model accuracy.

Protocols for Asynchronous Federated Learning

Due to their concurrent development, the algorithms proposed to perform asynchronous federated learning share many features in common. While in one light this represents a duplication of effort, in another it brings focus to those commonalities that are most promising to a next generation of techniques that represent more of a consensus among the community. The algorithm presented in (Sprague et al. 2019), unnamed in the body of the text but referred to as "Async" in figures, presents a good starting point as the algorithm most similar to FederatedAveraging. The main alteration is to the aggregation action of the server. Instead of waiting for the round of clients to return their parameters, the server in Async will immediately aggregate and return global parameters when it receives communication from a client. This change also introduces the concept of an aggregation weight (alternately called the mixing hyperparameter), a variable to control the amount of influence a newly-received local model has on the global model.

$$x_t \leftarrow (1-\alpha)x_{t-1} + \alpha x_{local} \tag{1}$$

In Equation 1, x_t represents the global model at time t, and x_{local} represents a newly-received model update from a client device, and α is the aggregation weight. Ranging from 0 to 1, a larger aggregation weight places more influence on the model updates communicated from the edge. Such a hyperparameter is uniquely valuable in an asynchronous protocol because it allows for fine-tuned control over the global model's tolerance to staleness (alternately called lag) - the age of the received local model compared to the current global model, whether that be by iterations, rounds of communication, or wall-clock time. Too much tolerance to stale local updates threatens the convergence of the global model, whereas a very low tolerance necessitates frequent communication that can eliminate the benefits of asynchrony.

Although (Sprague et al. 2019) identifies the need for an aggregation weight, the authors simply pick a static value of 0.1, citing that it strikes the right balance between stability and performance. (Xie, Koyejo, and Gupta 2019) explores the question further, introducing a similar but more nuanced protocol, dubbed FedAsync, that includes the ability to adaptively change the aggregation weight as a function of staleness. FedAsync provides further details on the internal workings of the server and client devices in the asynchronous setting, such as the non-blocking communication between multiple threads running on each device. Xie, Koyejo, and Gupta also list multiple ways that aggregation weight can be made a function of staleness, including linear, polynomial, exponential, and hinge functions. While lowering the impact of the client models via the aggregation weight does reduce the variance introduced by stale updates, it also slows the convergence of the global model. Factoring this into the trade-off between convergence and communication cost noted above yields a more complete picture; placing an emphasis on new updates from clients will cause the global model will converge faster, but also increases the destabilizing effect of stale parameters. At the same time, refreshing client models more frequently to reduce staleness increases the cost of communication.

The SAFA protocol introduced in (Wu et al. 2019) includes a unified "lag tolerance" hyperparameter τ to help manage the complex dynamic between staleness and convergence. Perhaps the most sophisticated algorithm of the group, SAFA, or Semi-Asynchronous Federated Averaging, has the sole distinction of being developed with knowledge of earlier asynchronous efforts. In this case, Wu et al. remark on the complicated nature of FedAsync's hyperparameters as inspiration for their design. SAFA proceeds in three stages: model distribution with lag tolerance, "first come first merge" client selection, and discriminative aggregation. See Figure 1 for an overview of the process.

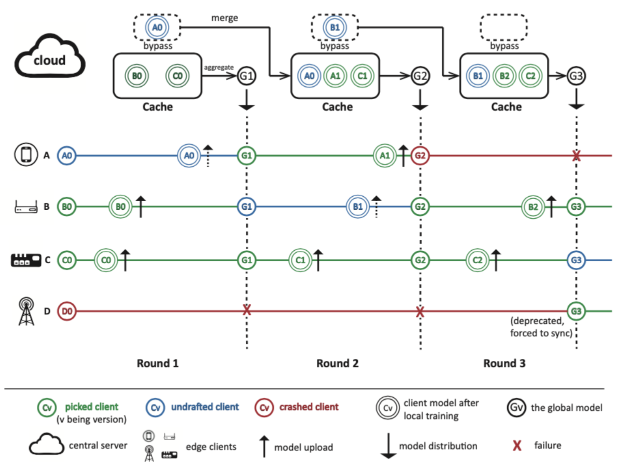


Figure 1: Diagram of SAFA protocol. (Wu et al. 2019)

In the model distribution stage, SAFA reduces communication overhead by only forcing clients to update their models when they are deemed "deprecated", meaning that their local model is behind the global version by more than lag tolerance τ. The next stage of operation addresses the straggler problem from synchronous federated learning by allowing all clients to participate, ending the current round when an approximate fraction of clients have reported back. Straggling clients are given more time to complete training without slowing down global progress, and other clients that were not selected in the current round are able to contribute their results. The selection algorithm prioritizes those clients that are less involved to avoid bias towards edge devices that have more favorable conditions. Finally, in the aggregation step, SAFA updates the global model and maintains a caching and bypass structure to allow client model updates that were not used in the current round to contribute to future aggregation. Via a parameter study, Wu et al. confirm the expected trade-off between communication overhead and loss as τ grows, concluding that a moderate amount of lag tolerance is optimal.

Standout Features

In addition to the general purpose algorithms presented in (Sprague et al. 2019), (Xie, Koyejo, and Gupta 2019), and (Wu et al. 2019), other researchers in the community have pioneered approaches with more narrow or unique goals and capabilities. In the case of (Chen, Ning, and Rangwala 2019) and (Li et al. 2019), these papers introduce asynchronous protocols that largely align with the general algorithms but include standout features or analysis. In the case of (Chen, Sun, and Jin 2019), the authors have a different conception of asynchronicity entirely.

Online Learning Along with the straggler effect, (Chen, Ning, and Rangwala 2019) addresses another shortcoming of baseline FederatedAveraging, support for online learning. Due to the high sample rate of on-board sensors on edge devices like smartphones, the amount of data on a client device can be actively increasing during the training process. The ASO-fed framework (asynchronous online federated learning) proceeds in a similar manner to Async, with the addition of dynamic learning step size on the server and online learning on the client. Dynamic step size is designed to account for inconsistencies resulting from model staleness, and works by increasing the learning step size on those client devices that contribute their models infrequently in hopes of achieving better learning importance. Client-side online learning is central to ASO-fed, and involves a novel online learning approach with an exponential moving average to place greater significance on data points that have arrived most recently.

Differential Privacy Concerned with the potential for membership inference or model inversion attacks against even asynchronous federated learning algorithms, (Li et al. 2019) explores the addition of differential privacy to an asynchronous protocol. The multi-stage adjustable private algorithm, or MAPA, builds differential privacy into an asynchronous algorithm that shares a lot in common with FedAsync, including a queue structure the server uses to immediately respond to client updates and a delay variable τ to place a bound on staleness. The addition of differential privacy is designed to ensure sample-level protection, and includes adaptive gradient clipping to reduce the total amount of noise needed and thus improve accuracy. The intuition, illustrated in Figure 2, is that gradient norm decreases in the learning process, making a static clipping threshold inadequate. To address this, MAPA adjusts learning rate to ensure the gradients norm decreases in stages, allowing for the gradual reduction of global sensitivity and thus noise.

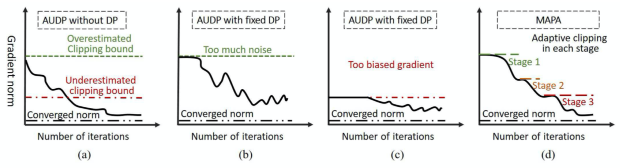


Figure 2: MAPA dynamic clipping bounds compared to base asynchronous federated learning with differential privacy (AUDP) (Li et al. 2019)

Proof of Convergence Xie, Koyejo, and Gupta state they are first to propose asynchronous federated training with provable convergence. This claim is largely borne out, though only by a matter of months. (Xie, Koyejo, and Gupta 2019) presents theoretical prove of convergence for the FedAsync algorithm with both strongly and weakly convex problems, under non-IID data distributions. They also provide some mathematical backing to the trade-off between convergence rate and error caused by the aggregation/mixing weight, as discussed by several of the papers in this review. Meanwhile in the privacy context, (Li et al. 2019) proves that their simplified AUDP scheme (asynchronous update with differential privacy) can converge even in light of out-of-date client updates and the addition of noise for differential privacy, provided the gradient staleness is not too large.

Selective Parameter Sharing (Chen, Sun, and Jin 2019) takes a different approach to asynchronicity. Instead of relaxing the strictly-synchronized communication rounds of FederatedAveraging, Chen, Sun, and Jin propose updating the deep and shallow layers of a Deep Neural Network (DNN) at different rates in order to reduce communication costs. Harking back to the selective parameter sharing discussed in (Shokri and Shmatikov 2015) and elsewhere, this approach is motivated by the intuition that shallow layers of a DNN tend to learn more general features, while deep layers learn feaures specific to the input data. Therefore, a federated learning algorithm could save communication bandwidth by transmitting updates to shallow layers more frequently than updates to deep layers. While this technique is not asynchronous in the same sense as the others in this review, it is not mutually exclusive with those other protocols. Perhaps future work will combine a truly asynchronous federated learning approach with a selective parameter-sharing scheme, yielding an even greater reduction in communication costs.

Experimental Results

Asynchronous federated learning is both a wholly new area of research and a field seeking to ameliorate real-world shortcomings of live federated learning deployments. For these reasons alone, design and formulation of robust experiments is hugely important. (Xie, Koyejo, and Gupta

2019) compares the FedAsync approach to baseline FederatedAveraging, simulating asynchronous "staleness" by random sampling from a uniform distribution. While their evaluation shows that FedAsync converges faster than FederatedAveraging with the same communication overhead when staleness is slow (and at a comparable rate when staleness is larger), their experimental design falls short of capturing all of the purported benefits of asynchronicity, such as increased scalability and resilience to hardware failure.

(Sprague et al. 2019) conducts more robust evaluation, better motivating the unique advantages of an asynchronous approach. As seen in Figure 3a, Sprague et al. test the Async algorithm with edge devices joining after 1000 and 7000 communication rounds. For each scenario, they run the experiment with balanced/IID data and unbalanced data, finding that only in the worst case (late joining with non-IID data) did the model fail to converge. To demonstrate resilience to heterogeneous client devices, Sprague et al. also construct a scenario where 90 out of 100 edge devices have their processing speed reduced by 10 or 100 times. As seen in 3b, Async is resilient to both scenarios.

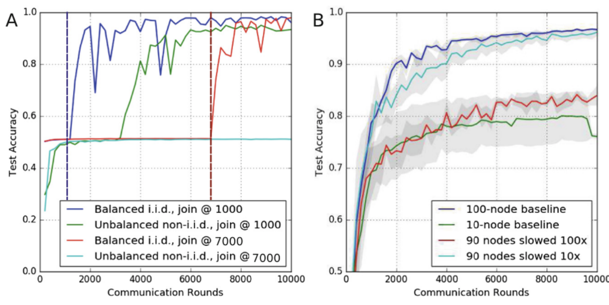
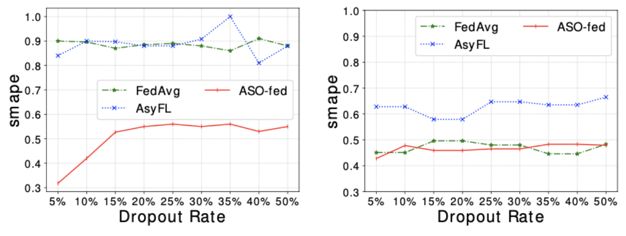


Figure 3: Affect of (a) edge devices joining late in the learning process and (b) edge devices with variable processing speeds (Sprague et al. 2019)

While (Sprague et al. 2019) demonstrates resilience to client "drop-in", (Chen, Ning, and Rangwala 2019) conducts experiments to simulate client drop-out. Across two real-world datasets, Chen, Ning, and Rangwala compare symmetric mean absolute percentage error (smape) of FederatedAveraging, a baseline asynchronous federated learning algorithm, and their proposed ASO-fed approach when the drop-out rate of client devices approaches 50%. As the results in Figure 4 reveal, the novel ASO-fed algorithm either exceeds or is competitive with FederatedAveraging, and outperforms baseline asynchronous federated learning due to its dynamic step size and online learning. Taken together, the experiments conducted over the papers in this review are a good first step towards validating the asynchronous approach, but leave the door open for more comprehensive evaluations in future work.

Conclusion

Having identified a shortcoming in a field that is itself emergent, asynchronous federated learning as a research community is in its infancy. Though young, it is very active and has strong theoretical underpinnings. Rigid, structured synchronization in the de facto algorithm designed to operate over a collection of devices that are by definition heterogeneous and prone to failure presents a fundamental contradiction that is ripe for exploration. Meanwhile, the research that inspired federated learning is far from settled on the topic of synchronicity, and the debate continues actively in the data center domain and elsewhere. With these two considerations in mind, it is clear that asynchronous federated learning has ample room to grow. It will be fascinating to see the refinements and new directions that subsequent works will take as the community enters a "second round of communication", building on the progress of these initial studies.



(a) FitRec (b) Air Quality

Figure 4: Affect of model drop out on FederatedAveraging, baseline asynchronous federated learning, and ASO-fed (Chen, Ning, and Rangwala 2019)

References

Balcan, M.-F.; Blum, A.; Fine, S.; and Mansour, Y. 2012. Distributed Learning, Communication Complexity and Privacy. *arXiv e-prints* arXiv:1204.3514.

Brendan McMahan, H.; Moore, E.; Ramage, D.; Hampson, S.; and Agüera y Arcas, B. 2016. Communication-Efficient Learning of Deep Networks from Decentralized Data. *arXiv e-prints* arXiv:1602.05629.

Chen, Y.; Ning, Y.; and Rangwala, H. 2019. Asynchronous Online Federated Learning for Edge Devices. *arXiv e-prints* arXiv:1911.02134.

Chen, Y.; Sun, X.; and Jin, Y. 2019. Communication-efficient federated deep learning with asynchronous model update and temporally weighted aggregation. *ArXiv* abs/1903.07424.

Dean, J.; Corrado, G.; Monga, R.; Chen, K.; Devin, M.; Mao, M.; aurelio Ranzato, M.; Senior, A.; Tucker, P.; Yang, K.; Le, Q. V.; and Ng, A. Y. 2012. Large scale distributed deep networks. In Pereira, F.; Burges, C. J. C.; Bottou, L.; and Weinberger, K. Q., eds., *Advances in Neural Information Processing Systems 25*. Curran Associates, Inc. 1223–1231.

Konečný, J.; McMahan, B.; and Ramage, D. 2015. Federated Optimization:Distributed Optimization Beyond the Datacenter. *arXiv e-prints* arXiv:1511.03575.

Li, Y.; Yang, S.; Ren, X.; and Zhao, C. 2019. Asynchronous

Federated Learning with Differential Privacy for Edge Intelligence. *arXiv e-prints* arXiv:1912.07902.

Pathak, M.; Rane, S.; and Raj, B. 2010. Multiparty differential privacy via aggregation of locally trained classifiers. In Lafferty, J. D.; Williams, C. K. I.; Shawe-Taylor, J.; Zemel, R. S.; and Culotta, A., eds., *Advances in Neural Information Processing Systems 23*. Curran Associates, Inc. 1876–1884.

Shokri, R., and Shmatikov, V. 2015. Privacy-preserving deep learning. In *2015 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, 909–910.

Sprague, M. R.; Jalalirad, A.; Scavuzzo, M.; Capota, C.; Neun, M.; Do, L.; and Kopp, M. 2019. Asynchronous federated learning for geospatial applications. In Monreale, A.; Alzate, C.; Kamp, M.; Krishnamurthy, Y.; Paurat, D.; Sayed-Mouchaweh, M.; Bifet, A.; Gama, J.; and Ribeiro, R. P., eds., *ECML PKDD 2018 Workshops*, 21–28. Cham: Springer International Publishing.

Wu, W.; He, L.; Lin, W.; Mao, R.; and Jarvis, S. A. 2019. Safa: a semi-asynchronous protocol for fast federated learning with low overhead. *ArXiv* abs/1910.01355.

Xie, C.; Koyejo, S.; and Gupta, I. 2019. Asynchronous Federated Optimization. *arXiv e-prints* arXiv:1903.03934.

Zheng, S.; Meng, Q.; Wang, T.; Chen, W.; Yu, N.; Ma, Z.-M.; and Liu, T.-Y. 2016. Asynchronous Stochastic Gradient Descent with Delay Compensation. *arXiv e-prints* arXiv:1609.08326.