Wireless Broadcasting for Efficiency and Accuracy in Federated Learning

Jonas Wessner[*]

The University of Hong Kong,
Pokfulam, Hong Kong

E-mail: jwessner@connect.hku.hk

Federated learning is a recent approach to machine learning where clients locally train a global model based on their private data and then upload their model parameters to a cloud server for merging into a global model. While it offers great opportunities for privacy-preserving AI solutions e.g. for medical applications, several challenges remain, limiting its efficiency and effectiveness, namely communication cost and the model parameter divergence in the presence of non-IID training data. In this paper, we analyze the state of the art with respect to the aforementioned challenges. Considering the characteristics of modern wireless networks, we notice that there is a huge opportunity for leveraging wireless broadcasting in a hybrid system architecture comprising peer-to-peer subgroups and a hierarchical server infrastructure. We design a new protocol for federated learning where peers share gradient updates with other nearby peers via wireless broadcasting and globally exchange gradient updates via a hierarchical server network. This way, we benefit from the efficiency of wireless broadcasting to increase communication efficiency and decrease server involvement. Furthermore, the frequent exchange of gradient updates between peers allows us to better cope with non-IID training data. Our protocol serves as a pattern that can be fine-tuned using many recently published contributions in federated learning. The impact of our work is expected to become even stronger in future 5G+ networks because it benefits from high device density and mobility.

International Symposium on Grids and Clouds (ISGC2024)
24 -29 March, 2024
Academia Sinica Computing Centre (ASGC), Institute of Physics, Academia Sinica Taipei, Taiwan

[*]Speaker

1. Introduction

Machine learning is today's fastest-developing and most powerful technology, finding applications in nearly every domain of science and industry, such as natural language processing, visual object detection, autonomous driving, stock market prediction, medical applications, and many more [1, 1]. For machine learning to be effective, a large amount of high-quality training data is essential [2, 220]. At the same time, the number of Internet of Things (IoT) devices such as smart watches, voice control systems, air quality monitors, surveillance cameras, and much more is increasing rapidly [3, 1]. These devices, as well as smartphones, are equipped with high-quality sensors generating an enormous amount of real-world data which has the potential to feed machine learning models to make them more powerful than ever [4, 1]. Traditionally, to make use of the data for training machine learning models, the data is uploaded to a cloud computing center, where it can be used for offline learning. However, this approach has two main downsides. Firstly, the velocity and volume of the data streams generated by the devices pose a challenge to the bandwidth and storage capacity of even large data centers [5, 56]. Secondly, since the data is usually highly privacy-sensitive, for example, medical data recorded by users' smart watches or private text messages, even anonymous uploading of the data to a cloud center is not acceptable and in some cases illegal [6, 2][4, 2].

To tackle these challenges, Google proposed a new approach to distributed machine learning called *federated learning* in 2016 [4]. The basic idea of federated learning is to use a large number of devices to cooperatively train a common machine learning model while keeping the training data private to the generating devices. The system architecture for federated learning is illustrated in Figure 1. It comprises a server (or multiple for fault tolerance) and many clients, where the end devices that produce the training data take the role of the clients. The server stores a machine learning model, which is downloaded by the clients. The clients train the machine learning model based on their local data in synchronous rounds. After each round, the clients upload their model parameter deltas to the server where they get merged into a new model based on weighted averaging. The new model then gets distributed to the clients for training in the subsequent round. Since no raw training data leaves the end devices, this approach guarantees some degree of privacy, which can be enhanced by using techniques such as mix networks [4, 3]. Moreover, as opposed to the training data, model updates are ephemeral and can be deleted directly after they get applied to the model of the respective round [4, 1], lowering the concern about privacy-sensitive data being stored in data centers for a long time. At the same time, this approach has the potential to cope with the network bandwidth and storage limitations of data centers [7, 2]. While for batch processing all training data must be transmitted and stored in the data center for the duration of the training, for federated learning only the comparatively small model updates must be transmitted [4, 3] and can be deleted right after the completion of the respective round.

While federated learning is a very promising approach and finds application for instance in Google's Gboard next word prediction [8] and medical applications [9], vanilla federated learning suffers from various issues such as parameter divergence, communication overhead, privacy concerns due to reconstruction of input data, single point of failure, malicious model updates, and the lack of computational power at end devices. For this reason, federated learning has become a hot topic in

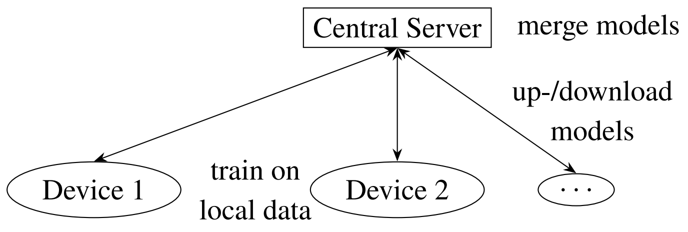

Figure 1: Illustration of federated learning showing (1) a server distributing a global model, (2) clients downloading the model and training it on local data, (3) the clients uploading the parameter deltas to be merged into a global model on the server.

recent years and many researchers proposed new ideas to optimize the federated learning approach. The remainder of this paper is organized as follows. In Section 2, we give an overview of the current challenges in federated learning and recent related contributions. After that, in Section 3, we propose a new protocol for federated learning that makes use of wireless broadcasting in local sub-networks to improve training convergence and reduce communication costs. Finally, we discuss future research topics and give a conclusion in Sections 4 and 5.

2. Recent Challenges and Contributions in Federated Learning

Since federated learning was initially introduced by Google in 2016, it received great attention from researchers. While it yields great opportunities to empower machine learning with the immense privacy-sensitive data available, there are still challenges to using federated learning effectively in practice. In this section, we explain current challenges related to federated learning, namely the high communication cost, the dependence on a central server, and the handling of non-independent and identically distributed (non-IID) data. For all of the challenges, we give an overview of recent publications addressing the respective issue.

2.1 Minimizing Communication Cost

Communication cost is one of the main limiting factors of federated learning [5, 62-64][10, 1]. Since the clients have to send gradient updates to the server in each communication round, the cost of communication is linearly dependent on the training time and the number of participating devices, which both can be massive and are expected to grow in the future with the trend towards larger models, more training data [11, 859], and more IoT devices [3, 1]. Since the bandwidths of cloud computing centers are limited, communication hinders the scalability of federated learning. Furthermore, transmitting data consumes power, which is especially scarce at end devices.

To overcome the bandwidth limitations at the cloud server for federated learning, hierarchical federated learning has been proposed [12]. In this approach, a third layer of edge servers is inserted between the clients and the server. The clients synchronize frequently with their respective edge servers, which then occasionally synchronize with the other edge servers via the central cloud server. Thus, the load on the server can be reduced as in classical multi-tier architectures while still allowing the clients to get frequent input from other clients to prevent strong model parameter divergence.

Another approach to minimizing the communication overhead is by optimizing the ratio between local optimization rounds and global synchronization rounds. Given a resource constraint and considering the resources needed for local training and global aggregation, a recent publication [13] introduces an algorithm that optimizes resource consumption.

A further approach to reduce communication has been proposed by Yurochkin et al. [14], which uses a new algorithm for merging neural networks into a single higher-quality network. Other than in classical federated learning, the distributed models are not assumed to have the same initialization weights before local training and the merged model does not necessarily have the same hidden layer layout as the local models. With this approach, models that have been trained independently for a large number of rounds can be merged more effectively, thereby decreasing the required frequency of communication rounds.

Compression is a way to reduce the communication cost without interfering with the federated learning algorithm. It has been shown, for instance, how neural networks can be used for compression and decompression of gradient updates to reduce the data that has to be transmitted in each communication round [15]. Furthermore, many compression algorithms for generic use cases exist and are applicable to federated learning as well [16].

2.2 Eliminating the Central Server

In the original version of federated learning proposed by Google, a central server is needed for merging the gradient updates of the participating clients into a global model. The community is concerned about the dependency on a central server for several reasons. Firstly, the server is a single point of failure, meaning the whole system cannot operate in the case of unavailability of the server [17, 2,10][18, 2]. Secondly, handing control to a central server raises privacy concerns if the server is run by an untrusted third party [19, 150][18, 2][20, 2].

Salmeron et al. [17] conducted research on aggregation functions for federated learning in a peer-to-peer network. They chose a fully connected peer-to-peer network architecture, requiring every peer to send updates to every other peer in each algorithm step. This leads to linear state complexity and quadratic communication complexity with regard to the number of peers, which is much worse than the classical server-client setup. Nevertheless, this architecture is a way to improve fault tolerance by eliminating the server as the single point of failure.

Another peer-to-peer federated learning approach has been proposed in 2021 [19]. In their work, the authors put a focus on trust by proposing an algorithm for merging the peers' model parameters without sharing the raw parameters, utilizing an n-out-of-n secret sharing schema. Similar to the work published by Salmeron et al., they rely on a fully connected peer-to-peer network, which is not scalable in terms of communication cost and state complexity.

BrainTorrent, another peer-to-peer federated learning system with the goal of eliminating the dependency on a trusted single server, was proposed in 2019 [18]. Like the two aforementioned peer-to-peer approaches, it might be suitable for achieving more autonomy of the peers, but similar to the aforementioned approaches, it is unsuitable for large-scale wide-area networks because of its high communication cost.

An asynchronous decentralized communication scheme for federated learning was proposed by Bellet et al. [20]. In their system, every process works on behalf of a local clock and receives and processes updates from other peers asynchronously, allowing to eliminate the cost of synchronization, especially in the presence of stragglers. To provide stronger privacy guarantees, they add a Laplace noise to the exchanged parameters. The proposed system is not yet production-ready since it e.g. does not handle peers entering and exiting the network. Furthermore, it also suffers from immense communication costs due to communication in a fully-connected network. While the authors argue that broadcasting is efficient in wireless networks [20, 4], realistically the participants are unlikely to be in close enough proximity, and even if that would be required, the network's scalability would be limited to the devices within signal coverage.

2.3 Handling Non-IID Data

In federated learning, the clients use the locally available data for training. Since the data originates from the respective participant in the system, it is inherently biased. Considering, for example, next word prediction on digital keyboards, the frequency of the words used is different for every user because they talk about different topics and have different linguistic habits and backgrounds. Non-IID training data can negatively impact the training convergence due to model parameter divergence [21]. To cope with this issue, recently several contributions have been published.

Huang et al. tackled this challenge by clustering the training data with the K-means method and then training K models with federated learning [22], one for each cluster. For inference, data points first have to be mapped to a cluster to then use the respective model for prediction. While this results in multiple less general models, it effectively reduces the non-IID property of the training data per model, thereby increasing the convergence rate and prediction accuracy.

Zhao et al. have shown that there is not necessarily a clear boundary between federated learning and centralized learning. They show that by replicating a subset of the training data, a trade-off between privacy and accuracy can be achieved [21]. In certain scenarios where some users consent to sharing part of their data with the other system users, the prediction accuracy could be significantly improved by this technique.

In the work by Luping et al. from 2019, they dynamically measure a parameter trend in the training process and thereby are able to identify outliers that occur due to skewed data distribution [10]. Using this approach, they can handle non-IID data better and also reduce communication costs since updates based on outliers do not have to be transmitted to the server.

3. Wireless Peer-to-Peer Broadcasting for Federated Learning

In this section, we introduce our main contribution, a novel peer-to-peer protocol for enhancing federated learning in modern wireless networks.

3.1 Motivation

Wireless 5G and beyond networks are characterized by increasing device density and mobility due to ubiquitous IoT devices, autonomous driving, and passengers in vehicles. Having analyzed the current challenges and contributions in federated learning in Section 2, we ask the question of how we could use modern wireless networks to improve federated learning in terms of the two main challenges, communication cost and non-IID training data. We notice that there is a way to combine some of the aforementioned state-of-the-art techniques and add new ideas to build a protocol that can benefit from the characteristics of modern wireless networks. Our protocol uses wireless peer-to-peer broadcasting where it is efficient, namely in local environments, and a hierarchical server architecture in the wide-area context.

3.2 Protocol Description

Our protocol, illustrated in Figure 2, contains two classes of participants, peers and servers, where servers form an n-tier hierarchical structure. The peers communicate in peer-to-peer wireless broadcasting networks, determined by their location and transmission power. The number of servers in the bottom-most tier is significantly smaller than the number of peer sub-networks, such that multiple peer sub-networks are directly connected to the same server. Similarly, the number of servers in tier i is significantly smaller than the number of servers in tier $i-1$, such that many servers are connected to one of the higher-level servers until the hierarchy ends at a single server in tier n.

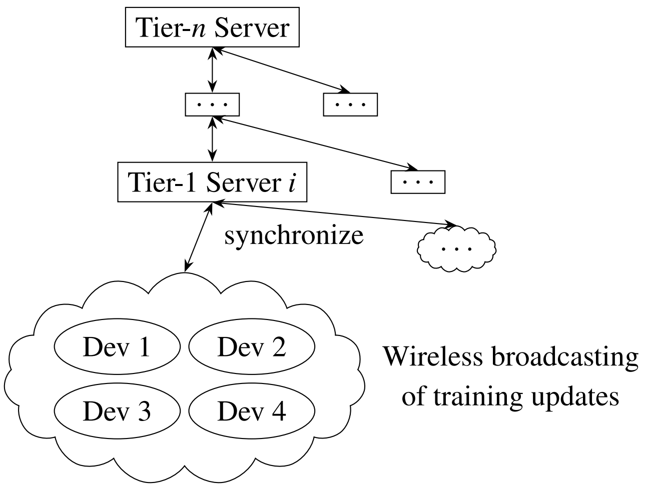

Figure 2: Illustration of our hybrid protocol for federated learning comprising multiple peer-to-peer networks connected through a hierarchical server infrastructure.

The protocol is described as follows:

1. Peers train their copy of the model on their local data in synchronous rounds. The length of a round is defined by a duration hyperparamter t. Whenever t time has elapsed, each peer broadcasts its model parameter weight deltas to nearby peers via a wireless broadcast channel.

2. At the end of each round, the peers wait to receive all parameter updates from nearby peers via the broadcast channel. After a predefined grace period, they assume that all nearby peers have finished sending (since the number of nearby peers is unknown and subject to change) and average those into a single model that they use for training in subsequent rounds. To account for different training speeds on different devices, they may use a weighted average concerning the number of training iterations done by the respective devices in this round.

3. Each peer i in each synchronization round j has a local timer that starts at a value $t_{s_i j} = t_s + t_{r_i j}$ and counts down, where t_s is a hyperparamter that defines the approximate duration between each synchronization of peer subgroups with the server and $t_{r_i j}$ is a pseudo-random number generated on peer i in for communication round j. When the timer of peer i expires, it resets its timer to a new value and broadcasts a message m_r, instructing all nearby peers to do so as well. This ensures a peer out of the group is randomly chosen to take action. Then all peers, except for peer i, send their most recent training updates to peer i analogous to step 1. This ensures that the model on peer i contains all recent updates of all other peers. Then peer i sends the gradient updates, representing the updates of all peers in the subgroup, to a tier-1 server.

4. Once a synchronization as described in step 3 has been initiated, all peers of this subgroup wait for the response of the server before continuing training.

5. Since the t_s value is the same across all peers, all peer networks will send an update to the tier-1 server at a similar time, only varying by the random component t_{ji}. The server waits for a predefined grace period to receive all the updates from peers since it does not know how many peer networks are connected to it because the peer networks can change due to user movement. After the grace period, the server averages the models and send the aggregated weights back to the peers that have sent the models.

6. Once peer i receives the aggregated model weights from the server, it broadcasts it to nearby piers, acting as a relay. All peers then continue training using this model as described in step 1.

7. After a predefined number of synchronizations with the server on a given tier, the servers of this tier (tier k) act as clients towards the tier $k + 1$ servers before replying to the tier $k - 1$. In this regard, the peers can be seen as a special tier 0.

8. If a peer does not have a local model (e.g. freshly entered the network), it behaves as if its $t_{s_i j}$ timer expired.

Note that the protocol already handles device mobility, as devices are loosely connected to other devices via a broadcasting channel. Furthermore, the edge case of a peer that does not have a local environment is considered. Such peer will never receive an m_r message and, for this reason, its timer will eventually run out. This causes the peer to synchronize with the server and behave like a 1-peer subgroup, similar to vanilla federated learning.

The number of tiers, the number of servers per tier, the timer values and the bounds of random numbers are variables that can be chosen based on the application circumstances. When implementing this simple protocol, many of the aforementioned state-of-the-art methods can be integrated: For the details on the multi-tier hierarchical server layout, the findings of Liu et. al can be used [12]. An approach similar to that of Wang et. al [13] can be used to determine the optimal bounds for the initialization of the peers' random timer. Of course, also advanced model merging schemes [14] and compression [15] can be used. Furthermore, any of the peer-to-peer architectures mentioned in Section 2.2 could be used to fine-tune the broadcasting within the local clusters. The techniques for handling non-IID data mentioned in section 2.3, namely sample clustering [22], sharing parts of the data [21] and global trend recognition [10] are also appropriate to be used in conjunction with this protocol.

3.3 Protocol Analysis

Our protocol design offers three major benefits for federated learning in wireless environments.

Firstly, compared to hierarchical federated learning, the average load on the server(s) is reduced by the factor of the average cluster size because only one peer per group communicates with a server in each step. Specifically, this means that given a certain surface area and transmission power, the communication cost of the system is constant with regard to the number of devices, allowing it to scale extremely well. This way, the architecture is perfectly suitable for super-dense 5G+ networks in big cities where numerous IoT devices and smartphones are located next to each other. The random selection of peers through the random timers ensures natural load distribution. This could be adjusted if a certain group of peers is desired to used as the relay. Furthermore, through the intelligent placement of the servers (e.g. placing bottom-most tier servers at the edge and letting peers communicate with nearby edge servers) low-latency and low-cost communication can be ensured.

Secondly, the frequent integration of nearby peers' model updates can improve the training convergence by reducing model parameter divergence. This is under the assumption that the data distribution is not strongly dependent on the location of the device. For instance, in next word prediction on keyboards, every user has specific topics and a specific way to express their thoughts and, hence, the training data is non-IID. However, if a couple of nearby users (e.g. on public transport or in a restaurant) share their model updates, the result of training is similar to when the data of all nearby devices would have been combined in one training data set. This effect gets stronger with high user mobility in 5G+ networks, since it makes the broadcasting groups more dynamic, i.e. one peer will be in the transmission range of a large variety of peers over time. This effectively increases the variety of data samples influencing each peer's model.

Thirdly, since the server(s), which are potentially hosted by an untrusted party, only see the combined gradient update of multiple peers transmitted by a random peer, better privacy protection is achieved at the server level. Even if the server was able to reconstruct information about the raw training data from the gradient deltas, it would be hard to associate this information with particular peers.

4. Future Work

Despite the relatively simple rules of the protocol, it leads to complex system characteristics that are worth being analyzed in more detail in the future:

It is not fully understood how the early migration through the broadcasting of training updates influences the convergence of the training process. While we anticipate that the increased interchange of model updates through moving clients randomizes the order of updates and thereby improves the convergence of federated learning, the practical effects of early migration are difficult to predict. There are three ways an update can be integrated into another model instance: by direct migration through broadcasting within the same subgroup, by early migration through moving peers and by downloading merged models from the server. The impact of those communication types is influenced by hyperparamters such as synchronization frequency and broadcasting transmission power, as well as by the movement patterns of the peers and the training data distribution. A simulation based on real-world mobile movement patterns and real-world distributed datasets could help to get more insight.

Another issue is the transmission power control for the broadcasting groups. In general, higher transmission power allows reaching more nearby peers, decreases the number of subgroups connected to the same server and increases the diversity of the shared training updates. The latter might also allow for less frequent synchronization with the server. Higher transmission power, however, comes at the cost of higher energy consumption, which is especially critical for mobile devices. Moreover, for a given objective, the optimal transmission power may vary among peers due to the non-uniform spatial distribution of peers.

How frequently to synchronize with the server and how frequently the servers should synchronize with upper-tier servers is also a relevant optimization problem. More frequent synchronization might aid in training convergence by mitigating model parameter divergence, but it also introduces communication overhead.

5. Conclusion

We have shown that the characteristics of 5G+ networks, such as high mobility and high device density, are a chance to develop loosely-coupled low-cost device-to-device architectures with interesting characteristics. In the context of federated learning, this can be useful to improve training convergence and reduce communication costs. Our hybrid protocol for federated learning is expected to become more powerful with the trend towards bigger cities, denser networks and more mobile IoT devices.

References

[1] M. I. Jordan and T. M. Mitchell, “Machine learning: Trends, perspectives, and prospects,” *Science*, vol. 349, no. 6245, pp. 255–260, 2015.

[2] R. C. Moore and W. Lewis, “Intelligent selection of language model training data,” in *Proceedings of the ACL 2010 conference short papers*, 2010, pp. 220–224.

[3] H. N. Saha, A. Mandal, and A. Sinha, “Recent trends in the internet of things,” in *2017 IEEE 7th annual computing and communication workshop and conference (CCWC)*. IEEE, 2017, pp. 1–4.

[4] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, “Communication-efficient learning of deep networks from decentralized data,” in *Artificial intelligence and statistics*. PMLR, 2017, pp. 1273–1282.

[5] Q. Yang, Y. Liu, Y. Cheng, Y. Kang, T. Chen, and H. Yu, *Horizontal Federated Learning*. Cham: Springer International Publishing, 2020, pp. 49–67.

[6] Q. Yang, Y. Liu, T. Chen, and Y. Tong, “Federated machine learning: Concept and applications,” *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 2, pp. 1–19, 2019.

[7] L. Li, Y. Fan, M. Tse, and K.-Y. Lin, “A review of applications in federated learning,” *Computers & Industrial Engineering*, vol. 149, p. 106854, 2020.

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

[9] S. Silva, B. A. Gutman, E. Romero, P. M. Thompson, A. Altmann, and M. Lorenzi, “Federated learning in distributed medical databases: Meta-analysis of large-scale subcortical brain data,” in *2019 IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019)*, 2019, pp. 270–274.

[10] W. Luping, W. Wei, and L. Bo, “Cmfl: Mitigating communication overhead for federated learning,” in *2019 IEEE 39th international conference on distributed computing systems (ICDCS)*. IEEE, 2019, pp. 954–964.

[11] T. Brants, A. C. Popat, P. Xu, F. J. Och, and J. Dean, “Large language models in machine translation,” 2007.

[12] L. Liu, J. Zhang, S. Song, and K. B. Letaief, “Client-edge-cloud hierarchical federated learning,” in *ICC 2020-2020 IEEE International Conference on Communications (ICC)*. IEEE, 2020, pp. 1–6.

[13] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, “Adaptive federated learning in resource constrained edge computing systems,” *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1205–1221, 2019.

[14] M. Yurochkin, M. Agarwal, S. Ghosh, K. Greenewald, N. Hoang, and Y. Khazaeni, "Bayesian nonparametric federated learning of neural networks," in *Proceedings of the 36th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. PMLR, 09–15 Jun 2019, pp. 7252–7261.

[15] H. Li and T. Han, "An end-to-end encrypted neural network for gradient updates transmission in federated learning," *arXiv preprint arXiv:1908.08340*, 2019.

[16] D. A. Lelewer and D. S. Hirschberg, "Data compression," *ACM Computing Surveys (CSUR)*, vol. 19, no. 3, pp. 261–296, 1987.

[17] J. L. Salmeron, I. Arévalo, and A. Ruiz-Celma, "Benchmarking federated strategies in peer-to-peer federated learning for biomedical data," *Heliyon*, 2023.

[18] A. G. Roy, S. Siddiqui, S. Pölsterl, N. Navab, and C. Wachinger, "Braintorrent: A peer-to-peer environment for decentralized federated learning," *arXiv preprint arXiv:1905.06731*, 2019.

[19] T. Wink and Z. Nochta, "An approach for peer-to-peer federated learning," in *2021 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*. IEEE, 2021, pp. 150–157.

[20] A. Bellet, R. Guerraoui, M. Taziki, and M. Tommasi, "Personalized and private peer-to-peer machine learning," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2018, pp. 473–481.

[21] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-iid data," *arXiv preprint arXiv:1806.00582*, 2018.

[22] L. Huang, A. L. Shea, H. Qian, A. Masurkar, H. Deng, and D. Liu, "Patient clustering improves efficiency of federated machine learning to predict mortality and hospital stay time using distributed electronic medical records," *Journal of biomedical informatics*, vol. 99, p. 103291, 2019.