Practical Federated Learning without a Server

Akash Dhasade
EPFL
Lausanne, Switzerland

Anne-Marie Kermarrec
EPFL
Lausanne, Switzerland

Erick Lavoie
University of Basel
Basel, Switzerland

Johan Pouwelse
Delft University of Technology
Delft, The Netherlands

Rishi Sharma
EPFL
Lausanne, Switzerland

Martijn de Vos
EPFL
Lausanne, Switzerland

Abstract

Federated Learning (FL) enables end-user devices to collaboratively train ML models without sharing raw data, thereby preserving data privacy. In FL, a central parameter server coordinates the learning process by iteratively aggregating the trained models received from clients. Yet, deploying a central server is not always feasible due to hardware unavailability, infrastructure constraints, or operational costs. We present Plexus, a fully decentralized FL system for large networks that operates without the drawbacks originating from having a central server. Plexus distributes the responsibilities of model aggregation and sampling among participating nodes while avoiding network-wide coordination. We evaluate Plexus using realistic traces for compute speed, pairwise latency and network capacity. Our experiments on three common learning tasks and with up to 1000 nodes empirically show that Plexus reduces time-to-accuracy by 1.4-1.6×, communication volume by 15.8-292× and training resources needed for convergence by 30.5-77.9× compared to conventional decentralized learning algorithms.

CCS Concepts: • **Computing methodologies → Distributed algorithms**; • **Computer systems organization → Peer-to-peer architectures**.

Keywords: Federated Learning, Decentralized Learning, Decentralized Peer Sampling

ACM Reference Format:
Akash Dhasade, Anne-Marie Kermarrec, Erick Lavoie, Johan Pouwelse, Rishi Sharma, and Martijn de Vos. 2025. Practical Federated Learning without a Server. In *The 5th Workshop on Machine Learning and Systems (EuroMLSys '25), March 30-April 3 2025, Rotterdam, Netherlands.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3721146.3721938

1 Introduction

Federated learning (FL) enables devices (referred to as *nodes* in this work) to collaboratively train a global machine learning (ML) model without sharing their private training data. In a single FL training round, a central server first selects a *sample*, *i.e.*, a random subset of online nodes, that train a model in parallel [41]. Nodes then send their updated model to the server, which aggregates incoming models into a single model. FL is widely used today in various applications, including next-word prediction on keyboards [13, 14, 20], speech recognition [53], human activity recognition [55], and healthcare [11, 38, 49, 52].

However, deploying and maintaining a central FL server can be challenging for various reasons [51, 65]. Infrastructure constraints in remote or underdeveloped areas may make it difficult to set up a reliable central server. Additionally, regulatory and privacy concerns might restrict centralized model aggregation, particularly in sensitive domains like healthcare and finance [44]. Furthermore, the reliance on a central server can introduce a single point of failure, stalling training progression if the server becomes unavailable [18, 28, 68]. Despite these challenges, centralized model aggregation used by FL offers an advantage over existing decentralized learning (DL) approaches that rely on neighborhood-based aggregation since FL results in higher model accuracy and faster model convergence. We thus ask ourselves the question: *Can we perform FL without a server?*

We answer affirmatively and present Plexus, a fully decentralized approach for FL training. The core of Plexus lies in its *decentralized peer sampler* that enables nodes to independently determine a subset of other nodes, or a *sample*, that is in charge of the training process for a given round. The sample changes every round, therefore evenly balancing the training load among nodes and providing nodes with an equal opportunity to contribute to model training. Following local model training, nodes in a sample select a single *aggregator* from the same sample that aggregates all trained models generated in each round. This aggregator then sends the aggregated model to nodes in the next sample, initiating the next round of training.

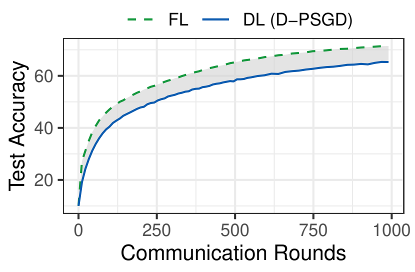

Figure 1. The evolution of test accuracy of FL and DL (D-PSGD) on the CIFAR-10 dataset in a 1000-node network. FL converges quicker and to higher accuracy than DL.

We evaluate Plexus using real-world mobile phone traces of pairwise latencies, bandwidth capacities, and computation speeds [32]. Our evaluation covers three common learning tasks in varying network sizes, up to 1000 nodes. We compare the performance of Plexus against standard FL and two baseline DL algorithms: decentralized parallel stochastic gradient descent (D-PSGD) [35] and gossip learning (GL) [21]. Our experimental results show that Plexus, compared to the best performing DL baseline, reduces time-to-accuracy by 1.4-1.6×, communication volume by 15.8-292×, and training resources consumed by 30.5-77.9×. Furthermore, we demonstrate that Plexus competes with the performance of FL in real-world settings.

This work makes the following two contributions:

1. We design Plexus, a practical and decentralized FL system (Section 3). Plexus incorporates a decentralized peer sampler to select a small subset of nodes that train the model each round, significantly reducing training resources required to converge compared to existing DL approaches. Our system operates without any centralized or network-wide coordination.
2. We implement Plexus and conduct an extensive evaluation of Plexus using real-world mobile phone traces at scale, comparing it with prominent baseline DL algorithms and standard FL with a server (Section 4). Our results demonstrate that Plexus, compared to DL baselines, significantly enhances performance across three common learning tasks regarding time-to-accuracy, communication volume, and resource requirements, while providing similar accuracy to FL.

2 Motivation

Federated learning (FL) is a collaborative ML algorithm where a central parameter server orchestrates the training of ML models on client devices. The effectiveness and practicality of FL has been well-documented in various settings [11, 13, 38, 49, 52, 53, 55]. However, its heavy reliance on a central server to coordinate training through *client sampling* and sample-wide *gradient aggregation* makes FL impractical in many real-world scenarios. In particular, FL training can last for days, and the central server needs to remain continuously available throughout the training process. Furthermore, the central parameter server in FL demands a high-bandwidth network connection to communicate with multiple clients simultaneously. These high availability and network bandwidth requirements result in significant operational and infrastructure costs for the FL infrastructure.

DL and Residual Variance. DL emerges as a promising alternative to FL [5]. DL approaches like D-PSGD [35], Epidemic learning (EL) [16], GL [46], and derivatives [6, 7] eliminate the need for the central server by introducing peer-to-peer communication and model aggregation. However, these DL approaches often do not attain the same accuracies as FL. This is because nodes in DL aggregate models amongst neighborhoods, *i.e.*, local aggregation [35]. Local aggregation leaves *residual variance* between local models, which biases gradient computations and slows down model convergence compared to performing a global aggregation before starting a training round [4, 29].

To further illustrate this effect, we chart in Figure 1 the test accuracy for FL and D-PSGD as the training progresses, on the CIFAR-10 dataset under an Independent and Identically Distributed (IID) data distribution in a network with 1000 nodes. We implement D-PSGD with the state-of-the-art one-peer exponential graph topology (OP-Exp.) [64]. With this topology, each node receives and sends exactly one model every round. A peer is connected to $log(n)$ neighbors (n is the total network size) and cycles through them round-robin. All other learning parameters follow our experimental setup described in Section 4.1. Figure 1 shows that FL's aggregation is beneficial for convergence and reaches higher accuracy than DL by nullifying residual variance across nodes in the network.

3 Design of Plexus

We first describe our system model and assumptions in Section 3.1, provide a conceptual overview of the algorithm in Section 3.2, and then present the components of Plexus in the remaining sections.

3.1 System Model and Assumptions

We consider a peer-to-peer network of n nodes that collaboratively train a global ML model θ. Each participating node has access to a local dataset which never leaves the participants' device. Only the model parameters are exchanged between participating nodes. We assume that each node knows the specifications of the ML model being trained, the learning hyperparameters, and the settings specific to Plexus. These specifications should be exchanged before training starts.

This work focusses on FL training in a cross-device setting without a central server. Thus, model training with Plexus proceeds in a decentralized environment and relies on the

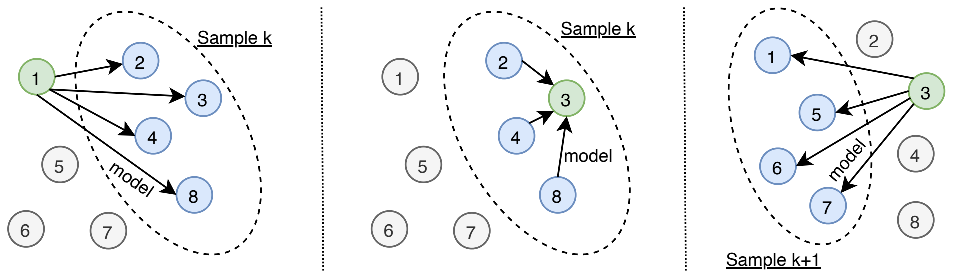

Figure 2. Overview of round k and $k + 1$ in Plexus, with 8 nodes and a sample size of 4 ($s = 4$). Participants are indicated in blue and the aggregator in green.

cooperation of nodes with varying resource capacities. We assume that each node has a unique identifier (*e.g.*, a public key) and assume that nodes are connected through a fully connected overlay network (*i.e.*, all nodes can communicate with each other). Though the computational capacities of participating nodes may vary, we assume that each node's computational resources are sufficient to reliably participate in learning. We assume that the model being trained fits into the memory of each node. Also, aggregators in Plexus should have sufficient memory or disk space to store and aggregate the trained models produced by other nodes. While we remark that nodes might act maliciously during the training process, we consciously leave out these considerations as this requires additional mechanisms. However, we acknowledge research in privacy-preserving ML, many of which we believe could be integrated or adapted into Plexus [6, 9, 42].

3.2 Plexus in a Nutshell

Similar to FL, Plexus *(i)* has a subset of nodes (a *sample*) train the model each round, and *(ii)* refreshes samples each round. We refer to nodes belonging to a sample as *participants*. Among the participants, one node, named the *aggregator*, is responsible for model aggregation during that round. This aggregator is selected to be the node with the highest bandwidth capacity as it has to temporarily handle incoming model transfers from all participants in a round. In each round, participants are randomly sampled from all nodes using a consistent hashing scheme. This sampling mechanism is a key contribution of Plexus and is further discussed in Section 3.3.

Figure 2 illustrates two rounds (round k and $k + 1$) in Plexus, with a network containing 8 nodes and a sample size of 4. We denote the set of nodes in the k-th sample as S^k and the aggregator within S^k as a^k. At the beginning of round k, the aggregator in the previous sample S^{k-1} sends the aggregated model to all participants in S^k (step 1). The participants train the model with their local data and then send their updated model to aggregator a^k (step 2). a^k finally sends the aggregated model to the participants in sample S^{k+1}, initiating round $k+1$ (step 3). This simplified algorithm description hinges on the ability of nodes to derive samples. Thus, the main technical challenge lies in deriving consistent samples in a decentralized fashion.

Algorithm 1 Determining a sample and aggregator by node i where k denotes the round number and s is the requested sample size.

```
1: procedure SAMPLE($k, s$)
2:     $H \leftarrow$ SORT([HASH($j + k$) for $j$ in NODES()])
3:     $C \leftarrow [j$ for $h_j$ in $H]$
4:     return $C[: s]$
5:
6: procedure AGGREGATOR($k, s$)
7:     $S^k \leftarrow$ SAMPLE($k, s$)
8:     return $j \in S^k$ such that $j$ has the largest bandwidth among
       all $S^k$ nodes according to $B_i$
```

3.3 Deriving Samples and Aggregators

One of the main novelties of Plexus is to decentralize the sampling procedure by having each participant in a sample compute the next sample independently. In order to achieve this, each node maintains a *local view* of the network wherein the membership information of (all) other nodes is recorded. The gist of Plexus's sampling procedure is to rely on a hash function parameterized by the round number and the node identifiers, stored by all nodes in their local view so that each node can independently compute the sample of nodes expected to be active during the training. Algorithm 1 shows the Plexus sampling procedure, which aims to obtain a sample of s currently active nodes in the k^{th} round. First, a subset of *candidates* is retrieved. Concatenating the node identifiers with round numbers randomizes the order of nodes every round. The resulting list is sorted in lexicographic order, which provides the order in which candidates are contacted. As long as the list of candidates is the same between all nodes, the order of contact and the resulting samples will mostly be the same.

Algorithm 2 Training and aggregation by node i.

```
Require: Sample size $s$, success fraction $sf$
$\Theta \leftarrow []$                          ▷ List of received models as aggregator

if $i$ in SAMPLE$(1, s)$ then   ▷ Start training if we are in the first sample
    send to $i$ train$(1,$ RANDOMMODEL$())$

upon train$(k, \theta)$                          ▷ Training
    $\bar{\theta} \leftarrow$ TRAIN$(\theta)$
    $a \leftarrow$ AGGREGATOR$(k, s)$                ▷ Alg. 1
    send to $a$ aggregate$(k, \bar{\theta})$

upon aggregate$(k, \theta_j)$ from $j$               ▷ Aggregation
    $\Theta$.ADD$(\theta_j)$
    if $\Theta$.SIZE $\geq \lfloor s \times sf \rfloor$ then
        for all $j$ in $S^{k+1}$ in parallel do
            $\theta_{agg} \leftarrow$ AVG$(\Theta)$
            send to $j$ train$(k+1, \theta_{agg})$
        $\Theta \leftarrow []$                      ▷ Reset list
```

We next discuss how participants determine an aggregator. Internally, this method calls the sampling procedure from Algorithm 1. The aggregator is a critical node for system progression and must handle the reception and transmission of at most s trained models during a round. Since the aggregator has to handle this network load, we preferentially select the participant with the highest bandwidth capacity from the derived sample, but acknowledge that other selection strategies are possible. This biased aggregator selection optimizes the model transfer times and reduces the time required per round compared to uniform aggregator selection. We remark that this biased selection has no impact on the quality of the trained model. Bandwidth capacities of individual nodes can be determined and synchronized a-priori to training. We found that this decision was essential to the success of Plexus as learning progress would slow down significantly if an aggregator with low bandwidth capacity is chosen, especially if the model size increases.

3.4 Training and Aggregating Models

Each node in Plexus implements two procedures: one for aggregation and one for training. We provide the pseudocode of these procedures in Algorithm 2. The design of Plexus is based on a *push*-based architecture, in which nodes in sample S^k trigger the activation of nodes in sample S^{k+1}. This way, nodes do not have to continuously be aware of the current training round being worked on; they only have to start working when receiving a trained or aggregated model.

Training. We first describe the training procedure by node i. A node starts the training task when it receives a `train` message containing an aggregated model θ and a round number k. i trains the received model θ by calling the TRAIN procedure, resulting in trained model $\bar{\theta}$. Then, i determines aggregator a in current sample k by calling the AGGREGATE procedure. This aggregator is derived using our peer sampler (see Section 3.3 and Algorithm 1). Finally, i sends an `Aggregate` message, containing the resulting model $\bar{\theta}$, to aggregator a.

Aggregation. We provide the pseudocode related to aggregation in Algorithm 2. An aggregator a starts the aggregation task when it is activated through an `aggregate` message from node j, containing trained model θ_j. a keeps track of the received models in list Θ and adds θ_j to Θ. Upon receiving at least $\lfloor s \times sf \rfloor$ models for round k, a aggregates these models, resulting in θ_{agg}. We refer to the required fraction of models needed as the *success fraction* sf. This oversampling is common in realistic FL systems [1]. a then determines the participants in sample $k+1$ and sends these nodes a `train` message containing the next round number $k+1$ and the aggregated model θ_{agg}. This completes round k.

3.5 Plexus and FL

Plexus brings FL to fully decentralized networks. We now discuss how the elements of Plexus translate to those of FL. Firstly, the local model training at the nodes in Plexus is equivalent to that in FL. The aggregator in Plexus temporarily performs the role of the FL server, aggregating the model updates in the current round. The central server in FL also orchestrates training by choosing the participants in each round. This role is handled by the decentralized sampling mechanism in Plexus described in Section 3.3. As a consequence of the similarity of Plexus to FL, the convergence proofs for FL also offer theoretical grounding for the convergence of our approach [34].

4 Experimental Evaluation

We now present the experimental evaluation of Plexus. We provide all relevant details on our experiment setup in Section 4.1. Our evaluation answers the following two questions:

- What is the performance of Plexus in terms of wall-clock convergence time, communication volume and training resource usage compared to FL and DL baselines (Section 4.2)?
- What is the effect of the sample size s in Plexus on wall-clock convergence time, communication volume and training resource usage (Section 4.3)?

4.1 Experiment Setup

We implement Plexus in the Python 3 programming language.[1] Plexus leverages the `IPv8` networking library which provides support for authenticated messaging and building decentralized overlay networks [59]. Our implementation adopts an event-driven programming model with the `asyncio` library. We use the `PyTorch` library [47] to train ML models, and the dataset API from DecentralizePy [17]. As

[1]Source code: https://github.com/sacs-epfl/plexus.

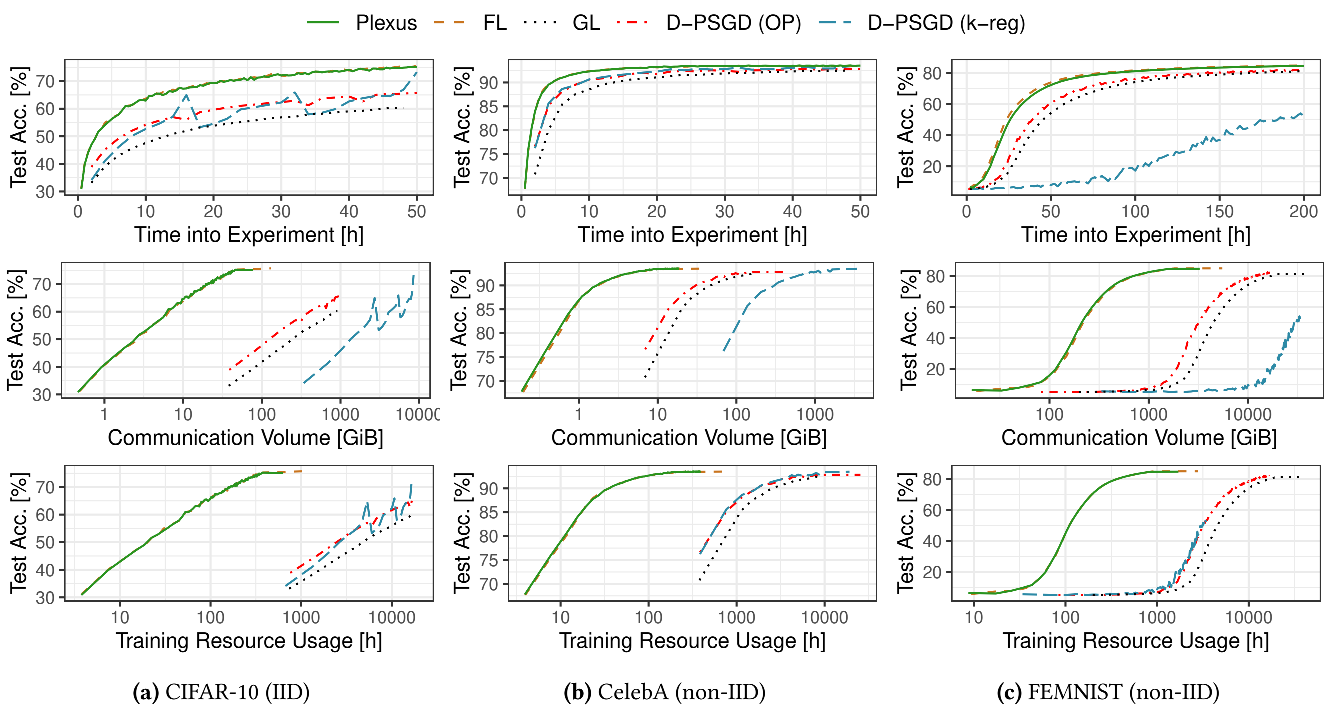

Figure 3. The convergence of Plexus and baselines against time (top), communication volume (middle), and training resource usage (bottom).

a node might be involved in multiple incoming and outgoing model transfers simultaneously, we equip each node with a bandwidth scheduler that we implemented. This scheduler coordinates all model transfers a particular node is involved in and enables us to realistically emulate the duration of model transfers.

Hardware. We run all experiments on machines in our national compute cluster. Each machine is equipped with dual 24-core AMD EPYC-2 CPU, 128 GB of memory, an NVIDIA RTX A4000 GPU, and is running CentOS 8. Similar to related work in the domain, we simulate the passing of time in our experiments [1, 32, 33]. We achieve this by customizing the default event loop provided by the `asyncio` library and processing events without delay. The real-world time required to reproduce our experiment depends on the baseline being evaluated. Running our Plexus and our FL baseline requires between 6 hours (for CelebA) and 24 hours (for FEMNIST) of compute time. The DL baselines are more computationally intense since they have every node training each round and require between 24 hours (for CelebA) and 5 days (for FEMNIST) of compute time. Our simulator is implemented as a single process and its efficiency can be further improved by paralellizing the training.

Traces. We have designed Plexus to operate in highly heterogeneous environments, such as mobile networks. To verify that Plexus also functions in such environments, we adopt various real-world traces to simulate pairwise network latency, bandwidth capacities, compute speed, and availability. To model a WAN environment, we apply latency to outgoing network traffic at the application layer to realistically model delays in sending Plexus control messages. To this end, we collect ping times from WonderNetwork, providing estimations on the RTT between their servers located in 227 geo-separated cities [62]. When starting an experiment, we assign peers to each city in a round-robin fashion and delay outgoing network traffic accordingly.

Table 1. Summary of datasets and learning parameters used to evaluate Plexus and DL baselines. "Mom." denotes the momentum parameter.

Dataset	Nodes	Learning Parameters	Model
CIFAR-10 [31]	1000	$\eta = 0.002$, mom. = 0.9	CNN [22]
CelebA [12]	500	$\eta = 0.001$	CNN
FEMNIST [12]	355	$\eta = 0.004$	CNN

We also adopt traces from the FedScale benchmark to simulate the hardware performance of nodes, specifically network and compute capacities [32]. These traces contain the hardware profile of 131 000 mobile devices and are originally sourced from the AI benchmark [26] and the MobiPerf measurements [23]. We assume that nodes are aware of the bandwidth capabilities of other nodes, and within a sample, a node sends its trained model to the aggregator with the highest bandwidth capability in the next sample. In summary, our experiments go beyond existing work on DL by integrating multiple traces that together account for the system heterogeneity in WAN environments.

Datasets. We evaluate Plexus on different models and with three distinct datasets, whose characteristics are displayed in Table 1. The CIFAR-10 dataset [31] is IID, partitioned by uniformly randomly assigning data samples to nodes. The CelebA and FEMNIST datasets are taken from the LEAF benchmark [12], which was specifically designed to evaluate the performance of learning tasks in non-IID settings. The sample-to-node assignment for CelebA and FEMNIST is given by the LEAF benchmark. Our evaluation, thus, covers a variety of learning tasks and data partitions.

Performance Metrics and Hyperparameters. We measure the top-1 test accuracy of the model on a global test set unseen during training, for the purpose of evaluation. In line with other work, we fix the batch size to 20 for all experiments and each device always performs five local steps when training its model [1, 32] before communicating. All models are trained using the SGD optimizer. For CIFAR-10 we additionally use a momentum factor of 0.9. All our learning parameters were adopted from previous works [4, 12] or were considered after trials on several values. They yield acceptable target accuracy for all evaluated datasets. We run each experiment three times with different seeds and report averaged values.

Baselines. We compare Plexus against a FL setup in which we assume the availability of a server with unlimited bandwidth constraints. We also use Gossip learning (GL) [21] and D-PSGD [35] as DL baselines. In each round of GL, a node first waits for some time and then sends its model to another random node in the network. The selection of nodes is facilitated by a peer-sampling service which presents a view of random nodes in the network every round. Upon receiving a model from another node, the recipient node merges it with its own local model, weighted by the model age, and trains the local model. GL naturally tolerates churn and is robust to failing nodes. However, pairwise model aggregation still leaves residual variance and deteriorates model convergence compared to when using global aggregation. In our experiments, we fix the round timeout to 60 seconds for GL to give each node sufficient time to train and transfer the model each local round.

D-PSGD [35] is a synchronous algorithm that only proceeds when all nodes have received all models from their neighboring nodes. We evaluate D-PSGD under two topologies: a 10-regular topology (*i.e.*, each node has ten neighbors) and a one-peer exponential graph topology, the latter being a state-of-the-art topology in DL [64]. Thus, we evaluate D-PSGD with sparse and dense graph connectivity.

For Plexus, we report the accuracy of the global model after aggregation every ten rounds. For D-PSGD, we determine the mean and standard deviation of the accuracy obtained by evaluating models of individual nodes on the test dataset every two hours. We also report communication volume (transmitted bytes) and training resource usage (*i.e.*, the time a device spends on model training). For Plexus, we set $s = 13$ and adjust the aggregator so it proceeds when it has received 80% of all models ($sf = 0.8$). We run experiments with CIFAR-10 and CelebA for 50 hours and FEMNIST for 200 hours which gives model training with Plexus and other baselines sufficient time to converge.

4.2 Plexus Compared to Baselines

We quantify and compare the performance of Plexus with baseline systems. We compare Plexus against a FL setup in which we assume the availability of *a server with unlimited bandwidth constraints*. We also use Gossip learning (GL) [21] and D-PSGD [35] as DL baselines. We evaluate D-PSGD under two topologies: a 10-regular topology (*i.e.*, each node has ten neighbors) and a one-peer exponential graph topology, the latter being a state-of-the-art topology in DL [64]. Thus, we evaluate D-PSGD with both sparse and dense graph connectivity.

Results. We show the performance of Plexus and baselines in Figure 3. The top row of Figure 3 shows the test accuracy as the experiment progresses. Plexus outperforms both DL baselines by converging quicker and achieving higher test accuracy, consistently across all datasets. In general, we find that in GL more training occurs within a given time unit compared to DL, since GL rounds are asynchronous and individual nodes have less idle time compared to D-PSGD. On the simpler binary classification task for the CelebA dataset, the performance improvements of Plexus are modest. However, on more difficult learning tasks like the 62-class image classification in FEMNIST with a larger model size, Plexus achieves more than 20% better accuracy when compared to the best performing DL baseline, GL. We also observe that Plexus generally shows comparable time-to-accuracy as FL but with the larger model size of FEMNIST we observe small differences since the server in FL has unlimited bandwidth. The middle row in Figure 3 shows the communication volume (horizontal axis in log scale) required to achieve the test accuracy for the evaluated systems. Plexus attains high test accuracies with orders of magnitude less transmitted bytes. These efficiency gains are particularly pronounced for the FEMNIST dataset. We note that D-PSGD with a 10-regular topology incurs the most network traffic while performing on par or worse than the one-peer exponential graph topology. The bottom row in Figure 3 shows the training resource usage (horizontal axis in log scale) consumed to achieve the test accuracy. Plexus attains high test accuracies with orders of magnitude less resource usage. Finally, we note that Plexus incurs comparable communication volume and resource usage as the centralized baseline of FL.

For each dataset, we determine the best-performing baseline in terms of the highest *individual model accuracy achieved across all nodes*. We then determine time-to-accuracy (TTA),

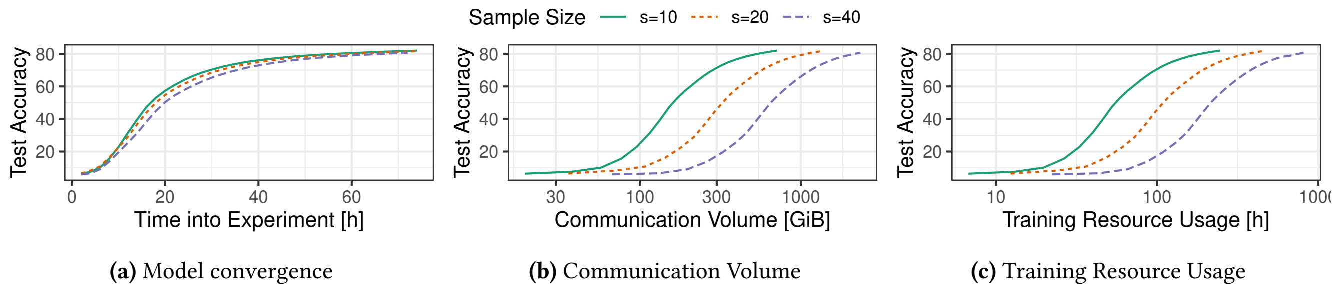

Figure 4. The performance of Plexus for different sample sizes s, using the FEMNIST dataset.

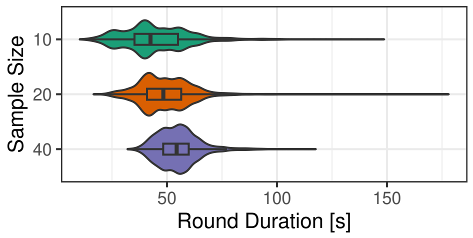

Figure 5. The distribution of round durations, for the FEMNIST dataset and different sample sizes.

communication-to-accuracy (CTA), and resources-to-accuracy (RTA), which are metrics that represent the efficacy, efficiency, and scalability of DL systems. For the evaluated datasets and compared to the target accuracy, *Plexus saves 1.4× - 1.6× in TTA, 15.8× - 292× in CTA and 30.5× - 77.9× in RTA compared to GL and D-PSGD.* In conclusion, our comprehensive evaluation demonstrates the superior efficiency and effectiveness of Plexus.

4.3 Varying the Sample Size s

We next explore the effect of the sample size s on model convergence, communication volume, training resource usage, and round duration by running Plexus with $s = 10$, 20, and 40. This experiment uses the FEMNIST dataset which has the largest model size in our setup. Figure 4a shows the test accuracy for the three different values of s as the experiment progresses. We observe that increasing s actually slows down training, likely because more models have to be transferred to and from the aggregator. Naturally, increasing s also has a negative impact on communication cost and resource usage, which are visualized in Figure 4b and Figure 4c, respectively. In particular, reaching 80% test accuracy for $s = 40$ incurs 4.0× additional communication volume and 4.0× more training resource usage, when compared to $s = 10$. We note, however, that increasing s might be favorable in other scenarios, *e.g.*, when data is highly non-IID.

Increasing s also prolongs the duration of individual rounds as the aggregator has to receive and redistribute more models. We show in Figure 5 the distribution of round durations in seconds for different values of s using a box and violin plot. When increasing s from 10 to 40, the average round duration increases from 45.1 seconds. to 54.9 seconds. Moreover, we observe that some rounds take disproportionally long which can happen when all nodes in a selected sample have low bandwidth or slow compute speeds. For example, a round for $s = 20$ took up up to 178 seconds. However, this is relatively rare: for $s = 20$, only 18 rounds out of 6941 took over 100 seconds to complete. We observe also a positive effect on round duration when increasing s: with higher values of s, there is a higher probability that nodes with high bandwidth capacities are included in the sample compared to lower values of s, which lowers the overall model transfer times during a round. We can see this effect in Figure 5 as there is a higher variance in round durations for lower values of s. Empirically, we obtain a good trade-off between sample size and convergence when setting the sample size around 10.

5 Related Work

Decentralized Learning (DL). D-PSGD [35] showed theoretically and empirically that under strong bandwidth limitations on an aggregation server in a data center, decentralized algorithms can converge faster. Assumptions on the behavior of those algorithms make them most suited to data centers. The synchronization required in D-PSGD is costly when training on edge devices. As a solution, research in DL has been focussed either on having a better topology [4, 16, 54, 56, 60, 64], or designing asynchronous algorithms [7, 37, 43, 46]. MoshpitSGD [50] uses a DHT to randomly combine nodes in multiple disjoint cohorts every round for fast-averaging convergence. Notably, Teleportation is a DL algorithm where, similar to Plexus, a small subset of nodes train every round and then exchange models with other sampled nodes over a smaller topology [57]. However, the paper does not specify exactly how these nodes are sampled and Plexus solves this issue through consistent hashing. All the above algorithms, however, overlook system heterogeneity whereas we evaluated Plexus in the presence of network and compute heterogeneity.

Federated Learning (FL). FL is arguably the most popular algorithm for privacy-preserving distributed learning and uses a parameter server to coordinate the learning process [41]. Similar to Plexus, FL lowers resource and communication costs at the edge by having a small subset of nodes train the model every round. To make FL suitable at scale in deployment scenarios, recent works have placed significant emphasis on system challenges [1, 8, 25, 33, 45, 66]. FL still requires a highly available central server that can support many clients concurrently, possibly resulting in high infrastructure costs. Plexus, on the contrary, is a fully decentralized system with an aggregation scheme inspired by FL, while avoiding central coordination.

Blockchain-Assisted DL. We identified various works that implement and evaluate blockchain-based decentralized learning systems [3, 30, 39, 40, 48, 61] and discuss the challenges therein [58, 67]. Consensus-based replicated ledgers used in these systems provide strong consensus primitives at the cost of a significant and unnecessary overhead. Machine learning optimizations based on SGD thrive in the presence of stochasticity, obviating the need for strong consensus in the form of a global model [36, 46].

Decentralized Peer Sampling. Brahms [10], Basalt [2], PeerSampling [27] and PeerSwap [19] provide each node with a different, uniformly random sample without network-wide synchronization. In contrast, the peer sampling of Plexus instead ensures that nodes select *equivalent* samples in a particular round of training.

6 Broader Impact and Open Challenges

The broader impact of our work is multifaceted, addressing both a technical and socio-economic dimension. By circumventing the need for a central server, Plexus can lower the barrier and costs to adopt FL-like training in decentralized settings. A complementary benefit of Plexus is that it enhances data privacy and security by decentralizing the model aggregation process, reducing the risk of data breaches associated with centralized systems. This is because, depending on the total network size and sample size, it becomes more difficult for a single node to systematically access model updates from particular nodes, raising the barrier for privacy attacks such as the gradient inversion attack [24].

Open Challenges. We discuss several open challenges in the current design of Plexus. Like any decentralized system, Plexus must balance efficiency with network constraints. As model sizes increase, aggregators may experience heavier loads, potentially prolonging round durations. This also applies to FL with a server, although it is typical that the server coordinating the FL process has more compute capacity than the devices participating in the training process. To account for this bottleneck in Plexus, we select the aggregator as the node with the highest bandwidth capabilities within a sample. However, in rare cases, all nodes in a sample may have low bandwidth capabilities due to an unfortunate selection (also see Section 4.3). In this scenario, we could select an aggregator from outside the current sample. Another approach to reduce communication burdens on aggregators is to use multiple aggregators in the same round. Both enhancements, however, require us to deviate from the standard FL algorithm and necessitate additional coordination mechanisms.

Secondly, we currently assume all nodes remain online during training which is typically not the case in real-world settings. Our idea is to extend Plexus with support for churn by including the current online or offline status in the local views, and synchronize these views between samples. However, dealing with churn requires additional coordination as nodes can also go offline during training or aggregation.

Thirdly, Plexus overutilizes the computational resources. In our experiments, the round completes when 80% of the models are received, thus at least 20% of trained models will never be aggregated. As a result, some participant updates may not be aggregated in every round, similar to common FL techniques that mitigate stragglers [8]. Various FL systems alleviate this issue by integrating stale model updates [1, 15, 63]. We leave integrating this in Plexus for future work.

Finally, securing Plexus against Byzantine nodes remains an open challenge. A promising direction is to incorporate accountability mechanisms where nodes verify the integrity of sample selection, ensuring honest participation.

Overall, these aspects highlight common challenges in FL and DL. Addressing them will improve the robustness and reliability of Plexus, paving the way for deployment of decentralized learning systems in open and large-scale settings.

7 Conclusions

This paper introduced Plexus, a practical, efficient and decentralized FL system. The two key components of our system are *(i)* a decentralized peer sampler to select small subsets of nodes each round, and *(ii)* a global aggregation operation within these selected subsets. Extensive evaluations with realistic traces of compute speed, network capacity, and availability in decentralized networks up to 1000 nodes demonstrate the superiority of Plexus over baseline DL algorithms, reducing time-to-accuracy by 1.4-1.6×, communication volume by 15.8-292×, and training resource usage by 30.5-77.9× compared to DL baselines. Moreover, Plexus also achieves accuracy and resource usage comparable to a centralized FL baseline.

Acknowledgments

This work has been funded by the Swiss National Science Foundation, under the project "FRIDAY: Frugal, Privacy-Aware and Practical Decentralized Learning", SNSF proposal No. 10.001.796. This work has also been funded by the Dutch national NWO/TKI science grant BLOCK.2019.004.

References

[1] Ahmed M Abdelmoniem, Atal Narayan Sahu, Marco Canini, and Suhaib A Fahmy. Refl: Resource-efficient federated learning. In *Proceedings of the Eighteenth European Conference on Computer Systems*, pages 215–232, 2023.

[2] Alex Auvolat, Yérom-David Bromberg, Davide Frey, and François Taïani. Basalt: A rock-solid foundation for epidemic consensus algorithms in very large, very open networks. *arXiv preprint arXiv:2102.04063*, 2021.

[3] Xianglin Bao, Cheng Su, Yan Xiong, Wenchao Huang, and Yifei Hu. Flchain: A blockchain for auditable federated learning with trust and incentive. In *2019 5th International Conference on Big Data Computing and Communications (BIGCOM)*, pages 151–159. IEEE, 2019.

[4] Aurélien Bellet, Anne-Marie Kermarrec, and Erick Lavoie. D-cliques: Compensating for data heterogeneity with topology in decentralized federated learning. In *2022 41st International Symposium on Reliable Distributed Systems (SRDS)*, pages 1–11. IEEE, 2022.

[5] Enrique Tomás Martínez Beltrán, Mario Quiles Pérez, Pedro Miguel Sánchez Sánchez, Sergio López Bernal, Gérôme Bovet, Manuel Gil Pérez, Gregorio Martínez Pérez, and Alberto Huertas Celdrán. Decentralized federated learning: Fundamentals, state of the art, frameworks, trends, and challenges. *IEEE Communications Surveys & Tutorials*, 2023.

[6] Sayan Biswas, Mathieu Even, Anne-Marie Kermarrec, Laurent Massoulié, Rafael Pires, Rishi Sharma, and Martijn de Vos. Noiseless privacy-preserving decentralized learning. *Proceedings on Privacy Enhancing Technologies*, 2025.

[7] Sayan Biswas, Anne-Marie Kermarrec, Alexis Marouani, Rafael Pires, Rishi Sharma, and Martijn de Vos. Boosting Asynchronous Decentralized Learning with Model Fragmentation. In *Proceedings of the ACM on Web Conference 2025*, 2025.

[8] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konečný, Stefano Mazzocchi, Brendan McMahan, et al. Towards federated learning at scale: System design. *Proceedings of machine learning and systems*, 1:374–388, 2019.

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

[10] Edward Bortnikov, Maxim Gurevich, Idit Keidar, Gabriel Kliot, and Alexander Shraer. Brahms: Byzantine resilient random membership sampling. *Computer Networks*, 53(13):2340–2359, 2009.

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

[12] Sebastian Caldas, Sai Meher Karthik Duddu, Peter Wu, Tian Li, Jakub Konečný, H Brendan McMahan, Virginia Smith, and Ameet Talwalkar. Leaf: A benchmark for federated settings. In *2nd Intl. Workshop on Federated Learning for Data Privacy and Confidentiality (FL-NeurIPS)*, 2019.

[13] Mingqing Chen, Rajiv Mathews, Tom Ouyang, and Françoise Beaufays. Federated learning of out-of-vocabulary words. *arXiv preprint arXiv:1903.10635*, 2019.

[14] Mingqing Chen, Ananda Theertha Suresh, Rajiv Mathews, Adeline Wong, Cyril Allauzen, Françoise Beaufays, and Michael Riley. Federated learning of n-gram language models. *arXiv preprint arXiv:1910.03432*, 2019.

[15] Georgios Damaskinos, Rachid Guerraoui, Anne-Marie Kermarrec, Vlad Nitu, Rhicheek Patra, and Francois Taiani. Fleet: Online federated learning via staleness awareness and performance prediction. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 13(5):1–30, 2022.

[16] Martijn De Vos, Sadegh Farhadkhani, Rachid Guerraoui, Anne-Marie Kermarrec, Rafael Pires, and Rishi Sharma. Epidemic learning: Boosting decentralized learning with randomized communication. *Advances in Neural Information Processing Systems*, 36, 2023.

[17] Akash Dhasade, Anne-Marie Kermarrec, Rafael Pires, Rishi Sharma, and Milos Vujasinovic. Decentralized learning made easy with decentralizepy. In *Proceedings of the 3rd Workshop on Machine Learning and Systems*, EuroMLSys '23, page 34–41, New York, NY, USA, 2023. Association for Computing Machinery.

[18] Anousheh Gholami, Nariman Torkzaban, and John S. Baras. Trusted decentralized federated learning. In *2022 IEEE 19th Annual Consumer Communications and Networking Conference (CCNC)*, pages 1–6, 2022.

[19] Rachid Guerraoui, Anne-Marie Kermarrec, Anastasiia Kucherenko, Rafael Pinot, and Martijn de Vos. Peerswap: A peer-sampler with randomness guarantees. In *2024 43rd International Symposium on Reliable Distributed Systems (SRDS)*, pages 271–281. IEEE, 2024.

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

[21] István Hegedűs, Gábor Danner, and Márk Jelasity. Gossip learning as a decentralized alternative to federated learning. In *Distributed Applications and Interoperable Systems: 19th IFIP WG 6.1 International Conference, DAIS 2019, Held as Part of the 14th International Federated Conference on Distributed Computing Techniques, DisCoTec 2019, Kongens Lyngby, Denmark, June 17–21, 2019, Proceedings 19*, pages 74–90. Springer, 2019.

[22] Kevin Hsieh, Amar Phanishayee, Onur Mutlu, and Phillip Gibbons. The non-iid data quagmire of decentralized machine learning. In *International Conference on Machine Learning*, pages 4387–4398. PMLR, 2020.

[23] Junxian Huang, Cheng Chen, Yutong Pei, Zhaoguang Wang, Zhiyun Qian, Feng Qian, Birjodh Tiwana, Qiang Xu, Z Mao, Ming Zhang, et al. Mobiperf: Mobile network measurement system. *Technical Report. University of Michigan and Microsoft Research*, 2011.

[24] Yangsibo Huang, Samyak Gupta, Zhao Song, Kai Li, and Sanjeev Arora. Evaluating gradient inversion attacks and defenses in federated learning. *Advances in Neural Information Processing Systems*, 34:7232–7241, 2021.

[25] Dzmitry Huba, John Nguyen, Kshitiz Malik, Ruiyu Zhu, Mike Rabbat, Ashkan Yousefpour, Carole-Jean Wu, Hongyuan Zhan, Pavel Ustinov, Harish Srinivas, Kaikai Wang, Anthony Shoumikhin, Jesik Min, and Mani Malek. Papaya: Practical, private, and scalable federated learning. In D. Marculescu, Y. Chi, and C. Wu, editors, *Proceedings of Machine Learning and Systems*, volume 4, pages 814–832, 2022.

[26] Andrey Ignatov, Radu Timofte, Andrei Kulik, Seungsoo Yang, Ke Wang, Felix Baum, Max Wu, Lirong Xu, and Luc Van Gool. Ai benchmark: All about deep learning on smartphones in 2019. In *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, pages 3617–3635. IEEE, 2019.

[27] Márk Jelasity, Spyros Voulgaris, Rachid Guerraoui, Anne-Marie Kermarrec, and Maarten Van Steen. Gossip-based peer sampling. *ACM Transactions on Computer Systems (TOCS)*, 25(3):8–es, 2007.

[28] Jiawen Kang, Zehui Xiong, Dusit Niyato, Yuze Zou, Yang Zhang, and Mohsen Guizani. Reliable federated learning for mobile networks. *IEEE Wireless Communications*, 27(2):72–80, 2020.

[29] Lingjing Kong, Tao Lin, Anastasia Koloskova, Martin Jaggi, and Sebastian Stich. Consensus control for decentralized deep learning. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 5686–5696. PMLR, 18–24 Jul 2021.

[30] Caner Korkmaz, Halil Eralp Kocas, Ahmet Uysal, Ahmed Masry, Oznur Ozkasap, and Baris Akgun. Chain fl: Decentralized federated machine

learning via blockchain. In *2020 Second International Conference on Blockchain Computing and Applications (BCCA)*, pages 140–146, 2020.

[31] Alex Krizhevsky. Learning Multiple Layers of Features from Tiny Images. https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf, 2009.

[32] Fan Lai, Yinwei Dai, Sanjay Singapuram, Jiachen Liu, Xiangfeng Zhu, Harsha Madhyastha, and Mosharaf Chowdhury. Fedscale: Benchmarking model and system performance of federated learning at scale. In *International Conference on Machine Learning*, pages 11814–11827. PMLR, 2022.

[33] Fan Lai, Xiangfeng Zhu, Harsha V. Madhyastha, and Mosharaf Chowdhury. Oort: Efficient federated learning via guided participant selection. In *15th USENIX Symposium on Operating Systems Design and Implementation (OSDI 21)*, pages 19–35. USENIX Association, July 2021.

[34] Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. On the convergence of fedavg on non-iid data. In *International Conference on Learning Representations*, 2019.

[35] Xiangru Lian, Ce Zhang, Huan Zhang, Cho-Jui Hsieh, Wei Zhang, and Ji Liu. Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent. *Advances in Neural Information Processing Systems*, 30, 2017.

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

[37] Xiangru Lian, Wei Zhang, Ce Zhang, and Ji Liu. Asynchronous decentralized parallel stochastic gradient descent. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 3043–3052. PMLR, 10–15 Jul 2018.

[38] Songtao Lu, Yawen Zhang, and Yunlong Wang. Decentralized federated learning for electronic health records. In *2020 54th Annual Conference on Information Sciences and Systems (CISS)*, pages 1–5. IEEE, 2020.

[39] Yunlong Lu, Xiaohong Huang, Yueyue Dai, Sabita Maharjan, and Yan Zhang. Blockchain and federated learning for privacy-preserved data sharing in industrial iot. *IEEE Transactions on Industrial Informatics*, 16(6):4177–4186, 2019.

[40] Umer Majeed and Choong Seon Hong. Flchain: Federated learning via mec-enabled blockchain network. In *2019 20th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, pages 1–4. IEEE, 2019.

[41] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-Efficient Learning of Deep Networks from Decentralized Data. In Aarti Singh and Jerry Zhu, editors, *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pages 1273–1282. PMLR, 20–22 Apr 2017.

[42] Viraaji Mothukuri, Reza M Parizi, Seyedamin Pouriyeh, Yan Huang, Ali Dehghantanha, and Gautam Srivastava. A survey on security and privacy of federated learning. *Future Generation Computer Systems*, 115:619–640, 2021.

[43] Adel Nabli, Eugene Belilovsky, and Edouard Oyallon. A2CiD2: Accelerating Asynchronous Communication in Decentralized Deep Learning. *Advances in Neural Information Processing Systems*, 36, 2023.

[44] Dinh C Nguyen, Quoc-Viet Pham, Pubudu N Pathirana, Ming Ding, Aruna Seneviratne, Zihuai Lin, Octavia Dobre, and Won-Joo Hwang. Federated learning for smart healthcare: A survey. *ACM Computing Surveys (Csur)*, 55(3):1–37, 2022.

[45] John Nguyen, Kshitiz Malik, Hongyuan Zhan, Ashkan Yousefpour, Mike Rabbat, Mani Malek, and Dzmitry Huba. Federated learning with buffered asynchronous aggregation. In Gustau Camps-Valls, Francisco J. R. Ruiz, and Isabel Valera, editors, *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, volume 151 of *Proceedings of Machine Learning Research*, pages 3581–3607. PMLR, 28–30 Mar 2022.

[46] Róbert Ormándi, István Hegedűs, and Márk Jelasity. Gossip learning with linear models on fully distributed data. *Concurrency and Computation: Practice and Experience*, 25(4):556–571, 2013.

[47] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[48] Shiva Raj Pokhrel and Jinho Choi. Federated learning with blockchain for autonomous vehicles: Analysis and design challenges. *IEEE Transactions on Communications*, 68(8):4734–4746, 2020.

[49] Nicola Rieke, Jonny Hancox, Wenqi Li, Fausto Milletari, Holger R Roth, Shadi Albarqouni, Spyridon Bakas, Mathieu N Galtier, Bennett A Landman, Klaus Maier-Hein, et al. The future of digital health with federated learning. *NPJ digital medicine*, 3(1):1–7, 2020.

[50] Max Ryabinin, Eduard Gorbunov, Vsevolod Plokhotnyuk, and Gennady Pekhimenko. Moshpit sgd: Communication-efficient decentralized training on heterogeneous unreliable devices. *Advances in Neural Information Processing Systems*, 34, 2021.

[51] Mikael Sabuhi, Petr Musilek, and Cor-Paul Bezemer. Micro-fl: A fault-tolerant scalable microservice-based platform for federated learning. *Future Internet*, 16(3):70, 2024.

[52] William Schneble and Geethapriya Thamilarasu. Attack detection using federated learning in medical cyber-physical systems. In *Proc. 28th Int. Conf. Comput. Commun. Netw.(ICCCN)*, volume 29, pages 1–8, 2019.

[53] Khe Chai Sim, Françoise Beaufays, Arnaud Benard, Dhruv Guliani, Andreas Kabel, Nikhil Khare, Tamar Lucassen, Petr Zadrazil, Harry Zhang, and Leif Johnson. Personalization of end-to-end speech recognition on mobile devices for named entities. In *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 23–30. IEEE, 2019.

[54] Zhuoqing Song, Weijian Li, Kexin Jin, Lei Shi, Ming Yan, Wotao Yin, and Kun Yuan. Communication-efficient topologies for decentralized learning with \$o(1)\$ consensus rate. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

[55] Konstantin Sozinov, Vladimir Vlassov, and Sarunas Girdzijauskas. Human activity recognition using federated learning. In *2018 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Ubiquitous Computing & Communications, Big Data & Cloud Computing, Social Computing & Networking, Sustainable Computing & Communications (ISPA/IUCC/BDCloud/SocialCom/SustainCom)*, pages 1103–1111. IEEE, 2018.

[56] Yuki Takezawa, Ryoma Sato, Han Bao, Kenta Niwa, and Makoto Yamada. Beyond exponential graph: Communication-efficient topologies for decentralized learning via finite-time convergence. *Advances in Neural Information Processing Systems*, 36, 2023.

[57] Yuki Takezawa and Sebastian U Stich. Scalable decentralized learning with teleportation. In *International Conference on Learning Representations (ICLR)*, 2025.

[58] Yuming Tang, Yitian Zhang, Tao Niu, Zhen Li, Zijian Zhang, Huaping Chen, and Long Zhang. A survey on blockchain-based federated learning: Categorization, application and analysis. *CMES - Computer Modeling in Engineering and Sciences*, 139(3):2451–2477, 2024.

[59] Tribler Team. Ipv8 networking library. https://github.com/tribler/py-ipv8.

[60] Thijs Vogels, Hadrien Hendrikx, and Martin Jaggi. Beyond spectral gap: the role of the topology in decentralized learning. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

[61] Leon Witt, Usama Zafar, KuoYeh Shen, Felix Sattler, Dan Li, Songtao Wang, and Wojciech Samek. Decentralized and incentivized federated

learning: A blockchain-enabled framework utilising compressed soft-labels and peer consistency. *IEEE Transactions on Services Computing*, 17(4):1449–1464, 2024.

[62] WonderNetwork. Global ping statistics. https://wondernetwork.com/pings. Accessed: 2022-05-12.

[63] Wentai Wu, Ligang He, Weiwei Lin, Rui Mao, Carsten Maple, and Stephen Jarvis. Safa: A semi-asynchronous protocol for fast federated learning with low overhead. *IEEE Transactions on Computers*, 70(5):655–668, 2020.

[64] Bicheng Ying, Kun Yuan, Yiming Chen, Hanbin Hu, Pan Pan, and Wotao Yin. Exponential graph is provably efficient for decentralized deep training. *Advances in Neural Information Processing Systems*, 34:13975–13987, 2021.

[65] Liangqi Yuan, Ziran Wang, Lichao Sun, S Yu Philip, and Christopher G Brinton. Decentralized federated learning: A survey and perspective. *IEEE Internet of Things Journal*, 2024.

[66] Feilong Zhang, Xianming Liu, Shiyi Lin, Gang Wu, Xiong Zhou, Junjun Jiang, and Xiangyang Ji. No one idles: Efficient heterogeneous federated learning with parallel edge and server computation. In *International Conference on Machine Learning*, pages 41399–41413. PMLR, 2023.

[67] Haoran Zhang, Shan Jiang, and Shichang Xuan. Decentralized federated learning based on blockchain: concepts, framework, and challenges. *Computer Communications*, 216:140–150, 2024.

[68] Hongyi Zhang, Jan Bosch, and Helena Holmström Olsson. Federated learning systems: Architecture alternatives. In *2020 27th Asia-Pacific Software Engineering Conference (APSEC)*, pages 385–394. IEEE, 2020.