Implicit Model Specialization through DAG-based Decentralized Federated Learning

Jossekin Beilharz*
Hasso Plattner Institute
University of Potsdam
Potsdam, Germany
Charité – Universitätsmedizin Berlin
Berlin, Germany
jossekin.beilharz@hpi.de

Bjarne Pfitzner*
Hasso Plattner Institute
University of Potsdam
Potsdam, Germany
bjarne.pfitzner@hpi.de

Robert Schmid*
Hasso Plattner Institute
University of Potsdam
Potsdam, Germany
Charité – Universitätsmedizin Berlin
Berlin, Germany
robert.schmid@hpi.de

Paul Geppert
Hasso Plattner Institute
University of Potsdam
Potsdam, Germany
paul.geppert@hpi.de

Bert Arnrich
Hasso Plattner Institute
University of Potsdam
Potsdam, Germany
bert.arnrich@hpi.de

Andreas Polze
Hasso Plattner Institute
University of Potsdam
Potsdam, Germany
andreas.polze@hpi.de

ABSTRACT

Federated learning allows a group of distributed clients to train a common machine learning model on private data. The exchange of model updates is managed either by a central entity or in a decentralized way, e.g. by a blockchain. However, the strong generalization across all clients makes these approaches unsuited for non-independent and identically distributed (non-IID) data.

We propose a unified approach to decentralization and personalization in federated learning that is based on a directed acyclic graph (DAG) of model updates. Instead of training a single global model, clients specialize on their local data while using the model updates from other clients dependent on the similarity of their respective data. This specialization implicitly emerges from the DAG-based communication and selection of model updates. Thus, we enable the evolution of specialized models, which focus on a subset of the data and therefore cover non-IID data better than federated learning in a centralized or blockchain-based setup.

To the best of our knowledge, the proposed solution is the first to unite personalization and poisoning robustness in fully decentralized federated learning. Our evaluation shows that the specialization of models emerges directly from the DAG-based communication of model updates on three different datasets. Furthermore, we show stable model accuracy and less variance across clients when compared to federated averaging.

KEYWORDS

decentralized machine learning, consensus protocol, blockchain, personalized federated learning

*These authors contributed equally to this research

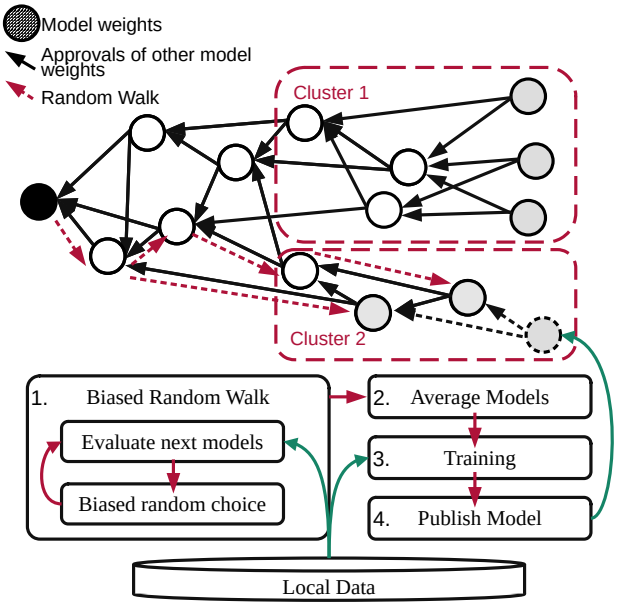


Figure 1: We use a biased random walk through a DAG of model updates to find models that perform well on local data, resulting in clusters emerging in the DAG.

Middleware '21, December 6–10, 2021, Virtual Event, Canada

ACM ISBN 978-1-4503-8534-3/21/12.
https://doi.org/10.1145/3464298.3493403

ACM Reference Format:
Jossekin Beilharz, Bjarne Pfitzner, Robert Schmid, Paul Geppert, Bert Arnrich, and Andreas Polze. 2021. Implicit Model Specialization through DAG-based Decentralized Federated Learning. In *22nd International Middleware Conference (Middleware '21), December 6–10, 2021, Virtual Event, Canada.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3464298.3493403

1 INTRODUCTION

Large datasets are often required in order to build powerful and accurate machine learning and especially deep learning models. This is problematic for many domains such as healthcare, since privacy regulations hinder sharing the sensitive data to create central databases. Instead, the data has to be used at the site where it was collected, resulting in multiple, sub-optimal, local models. As a solution, federated learning has been proposed, which allows many widely distributed nodes to train a common machine learning model on local data by exchanging updates to the model, but not the data itself. [30] The common model is usually exchanged via a central server, but the communication using decentralized consensus protocols like blockchains has been proposed. [7, 41, 53, 63] The model updates are stored on the chain and the participants jointly form a consensus by agreeing on the latest changes, thus defining the current global model parameters.

Another common challenge for federated learning and other decentralized learning approaches is the difference in data distributions present for different clients. [9, 24, 69] For this not independent and identically distributed (non-IID) data, model updates could counteract each other and hinder the training progress. [69]

In this paper, we demonstrate for the first time how the seemingly diverse goals of distributed model training, model personalization as well as robustness against poisoning attacks, can be addressed by a single mechanism that is inspired by distributed ledgers and federated averaging.

Specifically, we propose a fully-decentralized algorithm for solving a federated learning task, utilizing a model selection mechanism that incorporates the performance of other models on the local data: The so-called *accuracy tip selection* results in implicit model specialization for clusters of clients holding similar data. Using a synthetic dataset derived from MNIST, we demonstrate how the algorithm creates model convergence while still allowing specializations across different clusters. Furthermore, we show how malicious clients are isolated within the network, limiting their effects on other participants' models.

Additionally, we discuss means to tune the balance between generalization and specialization of local models as well as measures for the impact of this balancing parameter on cluster formation. In a quantitative evaluation, we compare the performance of our novel algorithm against centralized federated averaging for two more datasets, namely a dataset consisting of texts from William Shakespeare and Johann Wolfgang von Goethe, as well as CIFAR-100.

In this paper, terms that have different meanings in the context of distributed systems, graph theory, blockchains and federated learning are used. We use these terms with their following meaning: When using the term *node*, it is always specified whether a node of the graph or a compute node is meant. *Transaction* is only used in its meaning in the context of blockchains. In our approach each node of the directed acyclic graph represents a transaction, thus the terms "node of the graph" and "transaction" are used to describe different aspects of the same concept. *Consensus* is only used in its meaning in distributed systems and blockchains. *Weights* are used in three different contexts, which are clarified in each use: For the trained parameters of the machine learning model we specify them as "model weights". "Weights of transactions" or "weights of the random walk" are weights that are used in the random walk as part of the DAG-based consensus. Finally, "edge weights" are only used in Section 4.3 to talk about the modularity of a graph of clients that is different from the DAG that is used for consensus.

The remainder of this paper is structured as follows. The next section introduces the background to our work, while Section 3 discusses concrete examples of related work. Section 4 explains our approach and the implications for model performance, cluster identification, and poisoning robustness in detail. Section 5 presents the datasets and models that we used to evaluate our approach, as well as the results of the evaluation itself. Finally, Section 6 concludes this paper.

2 BACKGROUND

This paper builds upon research in the fields of federated learning and decentralized consensus mechanisms. Combining ideas from these fields, our approach is applied to the problem of multi-task learning and learning with non-IID data.

2.1 Federated Learning

Initially proposed in 2016, federated learning is a novel research area for developing machine learning methods for distributed datasets. [45] Federated learning relies on a client-server architecture, where the server defines a model to be trained, distributes the model weights to clients, and aggregates new model weights received back from clients. These are found after training the model for a number of epochs on the private, local client data. In this iterative fashion, the model is jointly improved until reaching optimal performance. In contrast to the field of distributed machine learning, the training facilitator does not have any control of, or access to client data. This brings privacy benefits for collaborators, since their data never leaves their location, but also entails problems with the training process if data is not independent and identically distributed. [69] Moreover, a key issue related to this is the communication cost, which becomes one of the main hinderances for quick convergence. Client devices may not always be available or have a bad internet connection in addition to potentially not having fast hardware for machine learning model optimization.

Recent works have suggested that the core goal of federated learning, to jointly develop a single model for all participants, is not ideal and has to be altered. [29] Instead, federated learning should also consider optimizing for the mean model performance per client, meaning that model personalization is desirable. A single model might not be able to optimally adapt to the non-IID nature of client datasets.

2.2 Permissionless Consensus

Consensus is a fundamental problem of distributed computing with a large body of research. Traditionally, consensus research focused on permissioned consensus, where the participating nodes of the consensus are known and authenticated to each other. Since the publication of Bitcoin [46] in 2008, there has been more research in permissionless decentralized consensus schemes. [3, 19, 31] In a blockchain, as introduced by Bitcoin, consensus on the ordering of transactions is reached. Network participants are selected

to propose the next block, often through a proof-of-work (PoW) mechanism. Participants then select the longest chain of blocks as the correct chain, thus abandoning forks that were introduced by other compute nodes at the same time. [46]

There are scalability limits of linear blockchains that lead to congestion in the network. [27] Different proposals have been made to solve these constraints. [4, 12, 42, 52] Recent distributed ledger technologies (DLTs), such as Byteball, Spectre, IOTA, Snowflake, and Conflux [10, 37, 55, 58, 61], have replaced the linear blockchain with a DAG. Similar to a blockchain, the DAG can be used to create consensus in a distributed system. The main goal of DAG-based DLTs is to allow for higher throughput of the system by allowing multiple concurrent blocks to be added simultaneously. Thus, while blockchains try to avoid forks and always select one winning chain, these forks are expected in DAG-based consensus mechanisms. Because forks are common, the question becomes now how the blocks are reintegrated to form the consensus. For this, each new block selects not only one, but multiple previous blocks to approve. Thus, over time, all valid blocks will be approved in the DAG.

Because of the potential for higher throughput, DAG-based consensus protocols are often seen as especially suited for use cases that involve large numbers of widely distributed resource-constrained devices like the internet of things, fog, and edge computing.

2.3 Multi-Task Learning and Non-IID Data

The domain of multi-task learning deals with solving multiple related tasks simultaneously by transferring knowledge between them during the training process. [6] For convex optimization problems such as training a linear or logistic regression model the challenge of multi-task learning often lies in finding a similarity matrix between tasks which is incorporated in the global optimization formulation. [68] An extension to multi-task learning called clustered multi-task learning further has the premise that some of the tasks are more related than others, forming clusters which all have similar model parameters. [28, 71]

For deep learning model architectures, multi-task learning is usually modeled as soft and hard parameter sharing between tasks. [56] In the former case, multiple models are learned, one per task, but the weights from different models influence each other, for instance by bounding the distance between them. The latter type re-uses some of the model layers completely, splitting the model at some point to end up with multiple prediction layers for the various tasks.

Federated learning with non-IID data can also be seen as a multi-task learning problem with the clients forming the different tasks. [11, 60] As simplification, clients can be clustered together if their datasets are similar and thus also their training task will be similar. Federated learning with cluster specialization is thus comparable to multi-task learning with soft parameter sharing.

3 RELATED WORK

The related work falls into two areas: decentralized federated learning, that is federated learning without a central server, and specialization in federated learning. Research in the area of decentralized federated learning relies on peer-to-peer networks to distribute the learning progress. One body of research focuses on the usage of blockchain architectures to create a global consensus on the learned model, while another, called *gossip learning*, relies on the participants themselves to merge models received from peers. While there is a significant interest in specialization or personalization in federated learning recently, this still is an emerging field of research.

3.1 Decentralized Federated Learning with DLTs

Many related works investigate the use of blockchains to communicate model updates in federated learning. [8, 32, 33, 59, 66, 67]

In these works, the gradient updates are inserted into the distributed ledger. The current network consensus on the global model is then defined by the model contained in the latest transaction that has a sufficiently high probability of being part of the longest chain of blocks.

Different manifestations of this general architecture have been proposed for different domains, such as health [35, 54], smart home [70] or railway operation [26].

For urban mobile networks, Lu et al. [41] address privacy and security concerns for federated learning. They propose a federated learning scheme where all nodes publish model updates and the corresponding mean absolute error. Interesting for our work is the use of a directed acyclic trust graph by Lu et al. to mitigate the risk of malicious nodes publishing wrong mean absolute errors, which would lead to biased averaging in their system. Schmid et al. [57] describe how a *learning tangle*, similar to the IOTA ledger, can be adapted for a decentralized, asynchronous implementation of federated averaging.

One topic in the research on federated learning in open networks is the question of fairness. That is, how to ensure the participants only benefit from the common machine learning model as much as they contributed to it. DeepChain [67] aims to guarantee fairness by a monetary incentive mechanism. Participants need to deposit monetary value, which is distributed to the other participants if a participant is found to be dishonest.

The FPPDL framework [43] transmits differentially private artificial samples and encrypted model updates via the blockchain. To ensure fairness, the mutual evaluation mechanism is based on points that can be earned by sharing model updates and then traded for the updates of other participants. The usage of local models instead of a global one is particularly relevant for this paper. However, in FPPDL these local models are only meant to restrict the model performance to be in line with the local contribution.

3.2 Decentralized Gossip Learning

Gossip learning algorithms [2, 15, 18, 21–23, 51, 64] also utilize a peer-to-peer network, but while blockchain approaches store learning progress on the ledger, this decentralized learning paradigm lets the participants deal with the way they include new information from their neighbors into their model. Clients periodically send out their current model paramters to a (randomly) selected peer. Upon receival of new parameters, a client merges their own model and the new one, for instance by taking the average, and updates the resulting model with their local data. Gossip learning research has investigated different sampling and merging algorithms to improve accuracy [13, 15, 22, 64], different compression schemes to improve

the communication efficiency [25], as well as different models such as GANs [21].

Hegedűs et al. [23] evaluated gossip learning against regular federated learning and found that for IID scenarios both approaches reach similar model performances. If the data is non-IID, however, gossip learning can struggle to converge quickly due to the lack of a central component. Looking at the communication efficiency, the authors claim that even though gossip learning requires more network traffic due to the peer-to-peer nature rather than having a single point of contact for all clients, the difference in convergence speed is relatively modest.

Some gossip learning research specifically investigates algorithms for creating personalized models. For instance, Vanhaesebrouck et al. [64] assume different training objectives per participant and propose two algorithms for incorporating knowledge from other people's models. Dinani et al. [15] put their focus on a dynamic network structure by example of a vehicular ad hoc network and incorporate the other models considering their marginal utility.

Compared to our approach, specialization for gossip learning can be reliant on the availability of related peers in the network which could slow down convergence speed [18]. Moreover, many gossip learning algorithms do not consider robustness against poisoning attacks. Even though recent work found, that peer-2-peer approaches may be more resilient than federated learning in real-world scenarios with many clients, they also state that more research is required to further investigate this topic [62].

3.3 Specialization in Federated Learning

One approach to allow for personalization in federated learning is to represent different client characteristics within a single global model. `FedProx` [39] generalizes and reparameterizes `FedAvg` to better account for both non-IID data distributions as well as stragglers that only submit partially trained updates into the synchronized averaging rounds. `Ditto` [38] defines fairness and robustness in federated learning and provides an updated optimization objective for the global model.

Other approaches use local models to improve the machine learning performance for individual clients. The benefit of this personalization has been shown for the language model of a virtual smartphone keyboard. [65] One personalized federated averaging algorithm, called Per-FedAvg [16] uses model-agnostic meta-learning [17] methods to find an initial shared model that users can adapt to their local data with comparably little training. Mansour et al. [44] propose three different approaches for the personalization of models: user clustering, data interpolation, and model interpolation. The user clustering is especially relevant for this paper. As part of this approach, hypothesis-based clustering is proposed to incorporate the machine learning task into the clustering. The idea of democratized learning [49, 50] aims to provide flexible distributed learning systems in which learning agents self-organize in a hierarchical structure to perform learning tasks together. Adaptive personalized learning [14] proposes the mixing of local and global models to form a personalized model that performs better than either the local or the global one.

4 APPROACH

In our approach of a Specializing DAG, the training runs in four steps for each client, as described in Figure 1: the biased random walk (1) selects two tips in the DAG, the models of these two tips are then averaged (2), the averaged model is improved by training (3) it on local data and, if the training improved the model, published (4).

This section describes our approach in detail. We discuss the mechanics of our DAG-based decentralized federated learning in two rounds: Section 4.1 explains the fundamental consensus mechanics applied to federated learning and Section 4.2 discusses in detail how the accuracy of foreign machine learning model updates can be incorporated. We then go on to explore its ability for implicit specialization and finally discuss implications for the robustness against poisoning attacks.

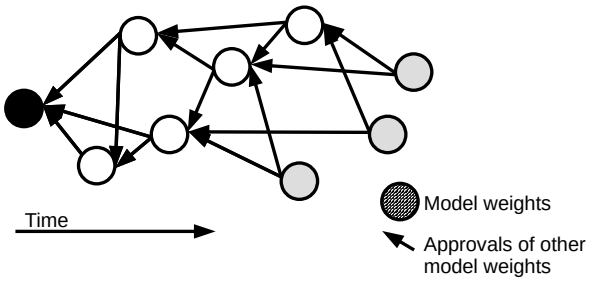


Figure 2: Communicating model updates in federated learning through a DAG: The nodes in the graph are model weight updates and the edges connect a weight update to the two other weight updates that were used as a basis for its training. Tips of the DAG (gray) are updates that didn't receive any approvals yet.

4.1 A DAG of Machine Learning Model Updates

We propose using a DAG of model updates for decentralized learning. Specifically, updated model weights are published as nodes in this DAG, while edges represent approvals of previous models that have been the basis of the current one, as illustrated in Figure 2.

Our DAG-based consensus is based on the approach of Popov [58], altered in a few key ways to adapt it to our use case of decentralized learning with implicit specialization. To publish a new model, a client averages two previously proposed models and trains the resulting averaged model on the local data. The new model update is then published on the DAG as approving the two models it was derived from. Clients only publish their model update if the training resulted in a model that performs better on the test data than the current consensus model.

Thus, compared to traditional federated learning, we propose to remove the central entity that averages the model weights and replace it with a decentralized consensus mechanism. Unlike many previous works on federated learning with decentralized models that communicate the model updates via a blockchain, using a DAG for the communication has two benefits. Firstly, the DAG allows for better scalability: multiple transactions can be proposed at the same time and still be reconciled because the newer transactions

approve more than one other transaction. Secondly, and more importantly, the structure of a DAG allows for more flexibility in the model communication which is harnessed to create an implicitly specializing system as discussed in the next section.

Our use of the DAG results in a few important differences to traditional DAG-based consensus. In DAG-based consensus mechanisms as used for cryptocurrencies, each transaction can be checked for consistency with other transactions. In our DAG of models, the transactions don't fall into the absolute categories 'valid' or 'not valid'. Rather, the quality of the model is a relative measure. A further difference in semantics of the DAG is that this quality of a model is dependent on the data the model should be applied to. When applied to federated learning with strong non-IID data, the quality of a model is very different for each client with its local data.

4.2 Enabling Implicit Specialization through Accuracy-Aware Tip Selection

In DAG-based consensus mechanisms, one important aspect is the tip selection algorithm - the algorithm that selects the tips that should be approved when publishing the next transaction. Schmid et al. [57] showed the general applicability of using DAG-based consensus for decentralized federated learning.

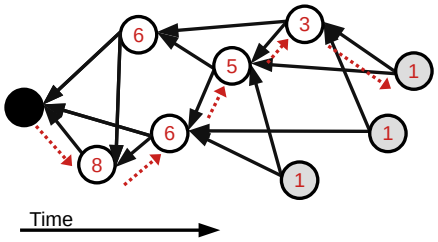


Figure 3: In DAG-based consensus schemes, traditionally weights of transactions are calculated by counting the number of approving transactions (also considering all transactions as self-approving). Thus, the weights are a global property of the DAG itself. The dashed red arrows show a random walk that always chooses the highest weight.

In this paper, we change this fundamental part of the consensus algorithm by integrating the model performance on the local data as bias of the tip selection algorithm. The tip selection algorithm is a random walk through the DAG in the opposite direction of the approvals. Traditionally, the random walk is biased by assigning each transaction a weight proportional to the size of the subgraph that spans behind it, as illustrated in Figure 3.

We change the bias of the random walk to be specific to a participating client. In each step during the walk, all potential next models, i.e. those reachable by taking one step in the DAG, are evaluated on the local test data, as shown in Algorithm 1.

The WeightedChoice chooses from these models randomly, weighted by the accuracies of the children on the local data. These accuracies are normalized by subtracting the maximum accuracy (see Eq. 1), resulting in negative normalized values. The weight

Algorithm 1 Random Walk of the Specializing DAG

```
procedure RandomWalk(Node n)
    children ← GetChildren(n)
    initialize accuracies
    for each child in children do
        accuracies_child ← EvaluateOnLocalData(child)
    nextNode ← WeightedChoice(accuracies, children)
    RandomWalk(nextNode)
```

between 0 and 1 is then calculated by taking the natural exponential of the normalized values scaled by a parameter α (see Eq. 2).

$$normalized = accuracy - \max(accuracies) \tag{1}$$

$$weight = \exp(normalized \times \alpha) \tag{2}$$

The amount of randomness in the walk can be determined by the α parameter, where higher values result in larger differences between the weights and thus less randomness and more determinism. Smaller values for α, on the other hand, lead to converging weights and thus more randomness. The expected differences in accuracies between models is dependent on the machine learning problem our approach is applied to, as well as the hyperparameters such as learning rate, batch size, and local epochs. In order to allow for good specialization even with small changes in accuracy between models and good generalization even with large differences, we include the spread of accuracies $\max(accuracies) - \min(accuracies)$ in each step as part of an altered normalized accuracy $normalized^*$:

$$normalized^* = \frac{accuracy - \max(accuracies)}{\max(accuracies) - \min(accuracies)} \tag{3}$$

We show the superiority of this altered normalization for certain values of α in our evaluation in Section 5.

This change in the tip selection of the consensus algorithm fundamentally changes the goal of the algorithm: from solely creating a consensus between the distributed clients on a central model to striking a balance between the generalization and specialization of the multiple client-local models.

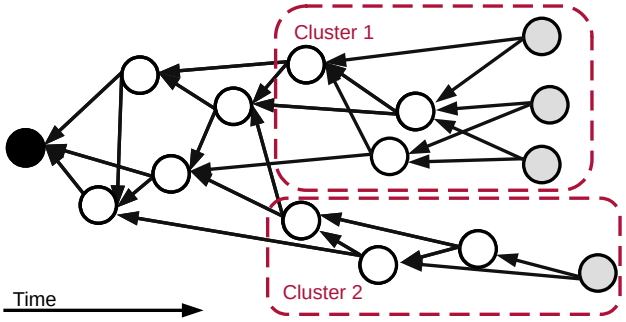


Figure 4: Tip selection using random walks biased by the model performance on local data leads to specialization and clustering in the directed acyclic graph.

4.3 Measuring Implicit Specialization

As described in the previous sections, the clients participating in the network implicitly form communities through mutual approvals of each other's transactions. Since the borders and memberships of these client communities are not directly expressed by the DAG, this section will introduce derived metrics for quantifying the degree of implicit specialization in the network.

The illustration in Figure 4 suggests that clusters in the DAG are visually easy to identify. However, in visualizations of actual experiment runs with many more model updates, this is not the case, especially since the randomness in the tip selection leads to frequent updates that connect two otherwise disjoint clusters. This is also the reason why finding the minimum cut within the DAG is not helpful in identifying large subgraphs that can be regarded as clusters.

Since in our experiments the set of participating clients is fixed and known in advance, we instead use a derived graph of clients $G_{clients}$ to identify communities: In this graph, the edge weight between two clients c_a and c_b is determined by the number of transactions that were published by c_a and directly approve a transaction of c_b or vice versa.

The *modularity* $m \in [-\frac{1}{2}, 1]$ is a measure for the existence of meaningful communities within a graph. Specifically, given a partitioning of the clients in $G_{clients}$, it expresses the difference between the accumulated edge weights within the partitions and the expected edge weights if they were randomly distributed among all clients in the graph. [47, 48] To obtain a fast approximation of the optimal graph partitioning achieving the highest modularity, we use the *Louvain* algorithm [1].

Since the accuracy-biased tip selection consistently selects models created by clients with similar data characteristics, the modularity of $G_{clients}$ should be positive for every DAG of model updates. Furthermore, once all clients have participated in training at least once, the edge weights within their communities are expected to continuously increase. Accordingly, the modularity should eventually converge to 1 during the course of our experiments.

As an additional measure of the specialization quality, we compare the clusters obtained by the Louvain algorithm with the clustering of clients that is known in advance. The *misclassification* fraction describes how many clients end up in a cluster where the relative majority of clients belongs to a different cluster according to the input labels.

The ability to identify groups of clients with similar characteristics can have negative implications on data privacy when participating in the network: Although clients publish their model updates anonymously, small clusters increase the potential for deanonymization attacks. Furthermore, if characteristics of a single client in a cluster are known, it may be possible to draw conclusions about private data of other clients that belong to the same cluster.

Section 5.3 discusses in detail how the previously introduced metrics can be used to optimize the random walk in the DAG.

4.4 Improved Robustness

For their decentralized learning DAG [57], Schmid et al. discussed two types of attacks that can be carried out to degrade the prediction accuracy for other participants in the network: Submitting random weights as well as weights that were trained using a mislabeled dataset, i.e. one in which labels of two classes are flipped. We adopt their threat model for this paper.

Submitting manipulated weights as a model update can prevent other nodes from publishing their training results because the final model does not improve, thus compute resources are wasted. Furthermore, if the malicious updates are dominating the network activity, they can take over the consensus, which leads to a degraded prediction performance for others.

With the accuracy-aware random walk, the effects of randomly generated model updates are effectively limited since their prediction accuracy is close to zero. Thus, given that attackers can only publish malicious updates at a limited rate, they must now make a compromise between poisoning effects and the probability that their transactions are selected by other clients during the biased random walk.

Furthermore, if attackers are aiming to influence the consensus model accuracy for the entire network, they would likely not use the accuracy-aware tip selection strategy as this would limit the effects of their attack to only a subset of the clients in the network. For targeted attacks at a cluster of clients, the success probability is increased since fewer transactions are necessary to dominate a subgraph of the DAG. Accordingly, for the sake of poisoning resistance, is is necessary to limit the degree of specialization by choosing a low value for the specialization parameter α.

In the remainder of this paper, we discuss a flipped-label attack scenario in which an attacker is able to manipulate the labels in the dataset of one or many clients, e.g. by installing forged sensing hardware. This means that attackers do not directly submit transactions into the network and cannot influence the tip selection process of the client. For the overall network, it is now desirable that (1) other clients remain unaffected from the data manipulation and (2) the affected clients are able to detect the data forgery.

In Section 5, we discuss if these objectives can be met using the proposed accuracy-aware random walk.

5 EVALUATION

We evaluated our approach on three datasets using a prototype implementation.

This section discusses the datasets and models used, and shows how the specialization emerges in the DAG for each of the datasets. Furthermore, we compare the performance of our approach against two other federated learning approaches on different machine learning tasks: Federated Averaging (*FedAvg*) is the original centralized federated learning process and *FedProx* is a state-of-the-art extension of FedAvg that accounts for non-iid data distributions amongst clients. Finally, we evaluate the poisoning resistance and scalability of our approach.

The prototype implementation of our approach and simulation used for this evaluation was published online.[1]

5.1 Datasets

We used three datasets with different characteristics to show the impact of our approach in different scenarios. Firstly, we evaluate a Handwriting recognition task on a synthetically clustered version

[1] https://github.com/osmhpi/federated-learning-dag

of the FEMNIST dataset, in addition to next character prediction on a new dataset from texts by Shakespeare and Goethe, and finally an image classification task on the CIFAR-100 dataset. Every client dataset has a train-test-split of 90:10.

5.1.1 FMNIST-Clustered. One of the most commonly used datasets for image classification is the MNIST dataset of 28x28 pixel handwritten digits [36] and its extension Extended MNIST (EMNIST) which also includes handwritten letters. For the federated case, the LEAF project [5] includes a dataset where the images are associated with the person who wrote the digits/letters (FEMNIST). To better show the effects of our approach, we opted for synthetically clustering clients by class, i.e. digit, and abandoning the split by author. Specifically, we constructed three disjoint clusters for classes {0, 1, 2, 3}, {4, 5, 6}, and {7, 8, 9} and assigned an equal number of clients to each cluster.

5.1.2 Poets. Our *poets* dataset is an extension of the *Shakespeare* dataset which is often used as a benchmark for federated learning. We evaluate the applicability of our approach to the task of next character prediction on this dataset. Poets combines texts from William Shakespeare and Johann Wolfgang von Goethe. The Shakespeare subdataset was preprocessed by the LEAF framework [5]. In addition, we extracted Goethe's plays from *Project Gutenberg* [20] and preprocessed them in the same way as the Shakespeare data. Both subdatasets were cleaned from clients with less than 1000 samples. To have an equal split between the number of English and German data samples, we reduced the Shakespeare data to 30% of its size by random sampling. We assigned the English and German datasets to separate clusters.

5.1.3 CIFAR-100. As image dataset with a non-synthetic clustering we investigated CIFAR-100 [34], including 32x32 pixel RGB images of different animals, objects or landscapes, belonging to 100 classes, which are each categorized into one of 20 superclasses. We use those superclasses as the clusters for our learning approach. The client data allocation was done using the Pachinko Allocation Method (PAM) [40] based on random draws (without replacement) from symmetric Dirichlet distributions over the superclasses and associated subclasses, as used by the Tensorflow Federated framework. In our experiments, all clients have both training and test data, which is required for calculating the weights of the random walk. We manually split each client's data into train and test partitions. In this dataset, clients possess data from more than one superclass/cluster, meaning there is no clear client-cluster affiliation. For analysis, we choose the cluster per client to be the most common one in its data, choosing randomly in case of a tie. Our CIFAR-100 dataset consists of 94 clients, the number of clients per cluster lies between three and six.

5.2 Models

The models for both the FMNIST-clustered and Poets dataset are based on models from the LEAF framework [5]. Prediction of FMNIST-clustered digits is done using a Convolutional Neural Net (CNN) with two ReLu activated convolutional layers with kernel size 5, and 32 and 64 filters, respectively, each followed by a max pooling layer with pool size and stride length 2. Afterwards, a fully connected layer with 2048 neurons and a ReLu activation function leads to the final fully connected layer with 10 output neurons and softmax activation for prediction.

For the Poets dataset, the LSTM model is fed an embedding of dimension 8, calculated from the 80 character sequence. The input is then fed through LSTM layers with 256 units each. Finally there is a dense output layer for prediction.

The classification model for CIFAR-100 is also a CNN similar to the one for FMNIST-clustered. After the two convolutional layers which are the same for both datasets, there is a third one with 128 filters, also followed by a max pooling layer. Finally, the model includes two fully-connected hidden layers and an output layer with 256, 128, and 100 neurons, in order.

The fixed training hyperparameters are shown in Table 1.

	FMNIST-clustered	Poets	CIFAR-100
Training rounds	100	100	100
Clients / round	10	10	10
Local epochs	1	1	5
Local batches	10	35	45
Batch size	10	10	10
Optimizer	SGD(0.05)	SGD(0.8)	SGD(0.01)

Table 1: Hyperparameters chosen for the evaluation. The number of local batches is fixed in order to equalize the number of batches used for training per client in case of an uneven distribution.

5.3 Results

We evaluated our approach with a prototypical Python implementation based on the work in LEAF [5]. For simplicity, we simulate the distributed training process in discrete rounds. We present our findings on three topics: optimizing the random walk by choosing good values for α for the FMNIST-clustered dataset, comparing the overall performance of our approach with federated learning as well as the poisoning robustness of our approach.

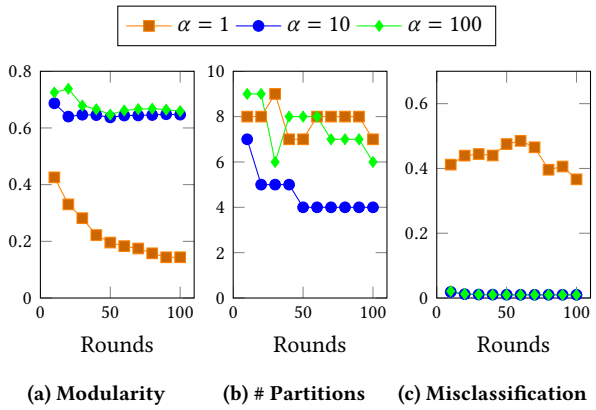
5.3.1 Optimizing the Random Walk. Section 4.2 describes how the α parameter controls the randomness involved in the biased random walk through the DAG. When configuring a decentralized learning task, it is necessary to determine a value of α that strikes a good balance between specialization and generalization for the learning task and non-IID data characteristics at hand. Specifically, we leverage the $G_{clients}$ graph and the metrics introduced in Section 4.3.

There are three criteria that indicate an appropriate choice of α: Firstly, the tip selection should be consistent enough so that in a majority of cases clients approve transactions only from other clients from the same cluster. This can be observed through the *approval pureness* metric. Additionally, the *modularity* metric of the $G_{clients}$ graph can show how clusters of clients emerge from the approvals without requiring pre-provided cluster labels.

Moreover, the corresponding partitioning of clients should consist of an appropriate number of partitions. If these are unreasonably many, α is set too high and the client models likely don't generalize

well. Finally, the model differences between clusters should become more distinct over time, which corresponds to a continuously decreasing misclassification fraction.

In our experiments, the approvals in the DAG show perfect pureness (approval pureness of 1.0) for $\alpha = 10$ and $\alpha = 100$, while $\alpha = 1$ shows less than half of the model updates approving other model updates from within the same cluster (approval pureness of 0.47). Still, the pureness for $\alpha = 1$ is higher than the 0.33 that would be expected for random approvals with three clusters.



(a) Modularity **(b) # Partitions** **(c) Misclassification**

Figure 5: Choosing α: On the FMNIST-clustered dataset, $\alpha = 10$ strikes the best balance with regards to the three metrics of $G_{clients}$.

Figure 5 shows the effects of choosing different values of α on the FMNIST-clustered dataset. Only a subset of 100 clients were included in these experiments, since distinct clusters within $G_{clients}$ can only be observed if the number of nodes in the graph is not continuously increasing. Nevertheless, the conclusions regarding the choice of α are valid also for experiments including a greater amount of clients.

A low value $\alpha = 1$ leads to decreasing modularity and performs poorly with regards to client similarities within clusters, as can be observed by the high fraction of misclassified clients. On the other hand, a high value $\alpha = 100$ also shows high and constant modularity, whereas the number of modules can be regarded as too high considering the three clusters in the training data. The medium value $\alpha = 10$ performs best: modularity is increasing slightly over time, there is a low number of modules and virtually all clients are assigned to a cluster corresponding to their label.

Besides evaluating the impact of of choosing the parameter α on the specialization, we also investigated its impact on the accuracy with the FMNIST-clustered dataset. Figure 6 shows the results for α values between 100 and 0.1.

For values of 10 and higher for α, the accuracy improves earlier than for values of 1 and lower. After 100 rounds, the accuracy still comes close to 1 for all values of α. For the lower values of α, no specialization emerges in the DAG for this dataset.

The good accuracy of the model after 100 rounds is due to the fact that eventually a generalized model learns to solve the task for all clusters. In the FMNIST-clustered dataset, the task is simple enough to solve for a generalized model.

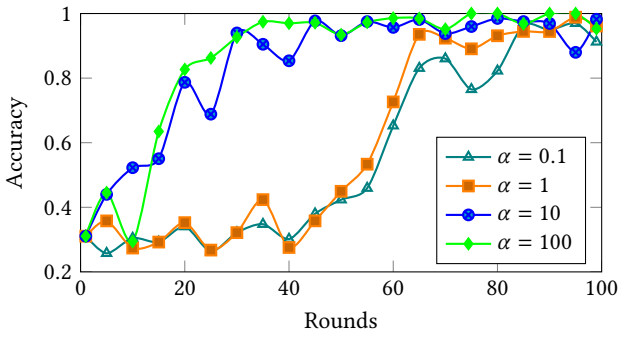


Figure 6: Higher values of α improve the accuracy for the FMNIST-clustered dataset.

While we would expect the accuracy to become worse for very high values of α this doesn't happen for *fully* clustered datasets (i.e. no class overlap between clients from different clusters) as there is no value in generalization.

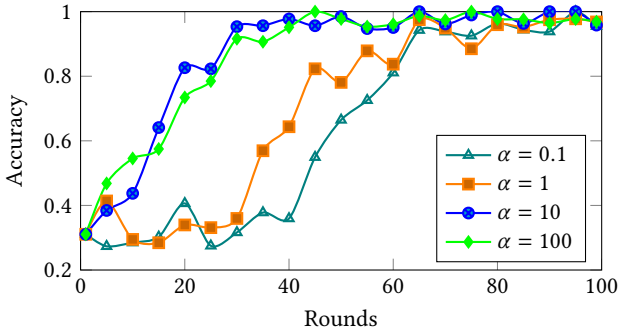


Figure 7: Choosing a dynamic normalization in calculating the weights of models during the tip selection results in better performance for $\alpha = 1$

The evaluation so far was done using the simple normalization calculation as explained in Section 4.2. Figure 7 shows the accuracy per round when using the altered normalized accuracy $normalized^*$ to calculate the weights in the random walk.

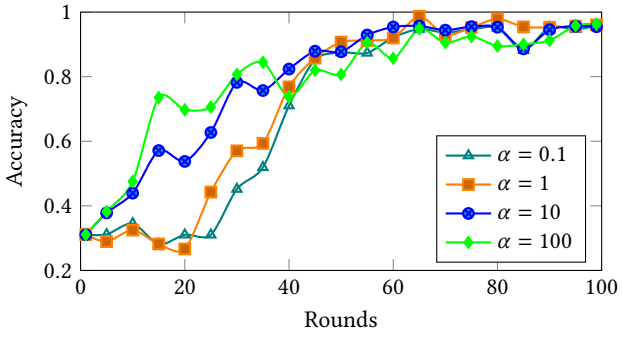


Figure 8: The effect that better accuracies are achieved faster remains with the relaxed dataset.

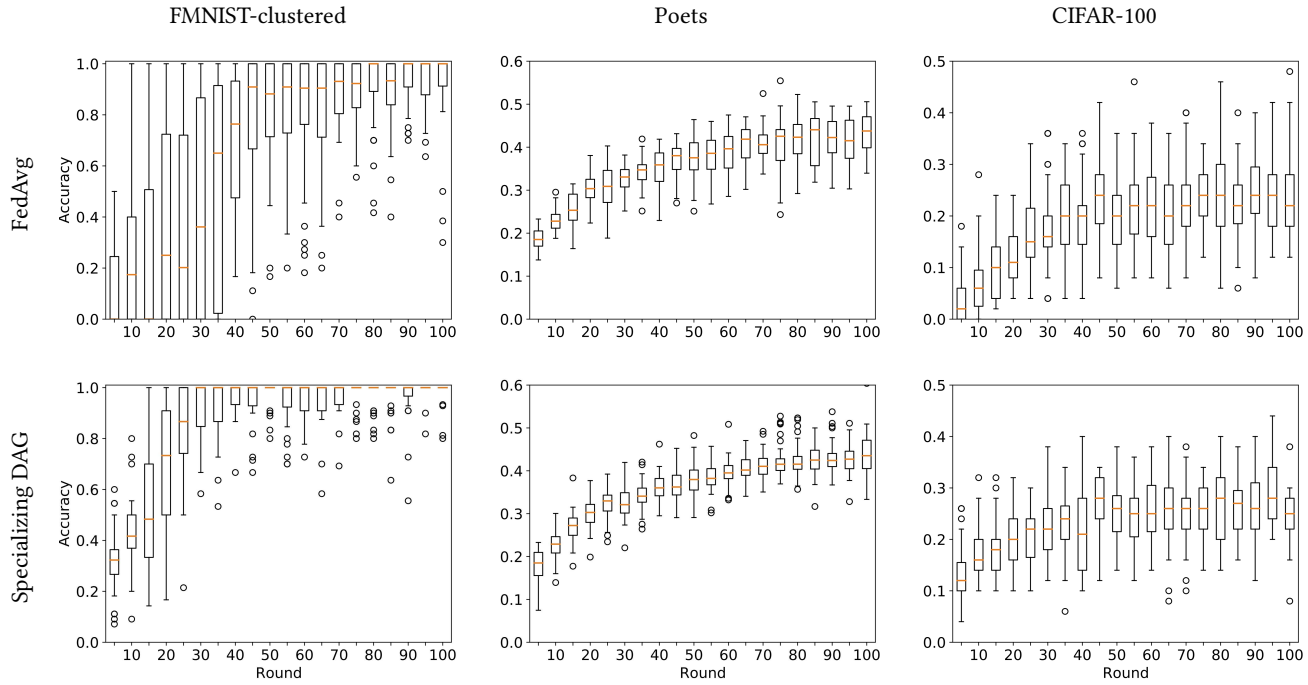


Figure 9: The accuracy on client-local data compared between federated averaging (FedAvg) and our approach (Specializing DAG), grouped over 5 rounds. FedAvg uses a central averaged model, while the DAG uses the specialized local models.

The usage of this altered normalization improves the accuracy slightly for $\alpha = 1$. This corresponds to a higher approval pureness of 0.51 for $\alpha = 1$ when using the dynamic normalization, compared to 0.40 with the standard normalization.

To evaluate the performance of our approach on not fully clustered data we created a relaxed FMNIST-clustered dataset, where each cluster contains between 15 and 20 percent of data from other clusters. Figure 8 shows the accuracy for different values of α for this relaxed dataset.

In general the relaxation of the clusters helps the model to generalize faster, resulting in better performance even for low values of α. At the same time, the performance of the well specialized cases with high values of α improve slightly slower because of the relaxation. Thus, while the same effect remains in this dataset, it is weaker than in the fully clustered dataset.

5.3.2 Comparison with Federated Averaging. We compare our approach with federated averaging as a baseline using the three different datasets. We first present the approval pureness of our approach for each dataset and then discuss the accuracy results.

Dataset	# clusters	base pureness	pureness
FMNIST-clustered	3	0.33	1.0
Poets	2	0.5	0.95
CIFAR-100	20	0.05	0.51

Table 2: The approval pureness in the DAG after 100 rounds of training with our approach.

To quantify how strong the DAG specialized in these experiments, we show the approval pureness in Table 2. In the FMNIST-clustered dataset there are three clusters for the groups of classes, Poets has the two clusters for texts by Goethe and Shakespeare and CIFAR-100 is clustered into the 20 superclasses. The base pureness is the approval pureness expected if the approvals would be randomly spread over all clusters.

The approvals in the DAG show perfect pureness for the FMNIST-clustered dataset. That is, all approvals of models are from within the same cluster, which is sensible for a completely clustered dataset where the integration of models from other clusters will not lead to better performance. For Poets and CIFAR-100, the approval pureness is lower and shows the balance between generalization and specialization into the clusters.

Figure 9 shows an overview of the accuracy of our approach compared to federated averaging. The values show the accuracy distribution on the local data of all clients selected in five consecutive rounds using the aggregated model in FedAvg and the locally optimized and published model for the Specializing DAG.

The accuracy evaluation shows that our approach performs better for the FMNIST-clustered dataset, where the accuracy improves faster. The larger deviation in federated averaging shows the missing ability to specialize. Consequently, the DAG is the first mechanism that enables training of a machine learning model on heterogeneous client datasets in a decentralized and asynchronous way.

For the Poets and CIFAR-100 datasets, the DAG achieves similar accuracy results compared to federated averaging. This also shows the feasibility of the decentralized approach: the central server can

be removed without an accuracy penalty for the evaluated datasets. Further improving the training accuracy on these datasets is an area of future work.

5.3.3 Comparison with FedProx. FedProx [39] is a state-of-the-art extension of FedAvg that guarantees model convergence even with non-IID client data distributions. The authors claim that in realistic scenarios, FedAvg only receives partial information from the clients, which it does not properly account for. These partial information can stem from statistical heterogeneity (non-IID data distributions) as well as stragglers, i.e. clients that were only able to submit partially trained models in time. Thus, the authors propose to add a proximal term to FedAvg that improves the convergence behavior in such heterogenous networks theoretically and empirically.

In the case of the specializing DAG, there are no stragglers due to its asynchronous nature: In a distributed implementation, each client continuously runs the training process as often as its resources permit, independent from all other clients. We only introduce the concept of rounds to be able to compare the performance of the DAG with centralized approaches.

For comparing ourselves with FedProx and unmodified FedAvg, we used the synthetic dataset proposed by FedProx: It is parameterized with $\alpha = 0.5$, $\beta = 0.5$, $\alpha, \beta \in [0; 1]$, where α and β control the dissimilarity of the local training samples for each client and between clients, respectively.

Figures 10 and 11 show the average accuracy and loss in a scenario with 30 clients in total and 10 active clients per round. The variance in accuracy and loss of the DAG is generally higher compared to the centralized approaches which can be explained by the statistical tip selection process as part of training and inference. However, the Specializing DAG eventually outperforms FedAvg in both accuracy and loss without the need for a central parameter server. Regarding the loss, the DAG results come close to the FedProx baseline, which shows how implicit specialization in the DAG effectively helps to accomodate differences in the data distribution among clients.

5.3.4 Poisoning. In order to investigate the robustness of the approach, we conducted experiments with flipped-label poisoning attacks using the original FMNIST dataset that is split by the authors

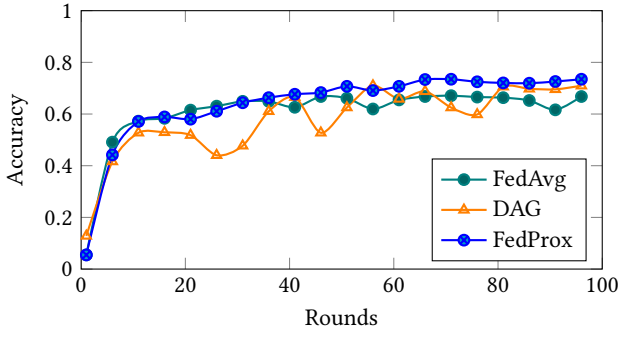


Figure 10: Initially, the average accuracy is more consistent using the centralized approaches; later, the DAG performance stabilizes and outperforms FedAvg.

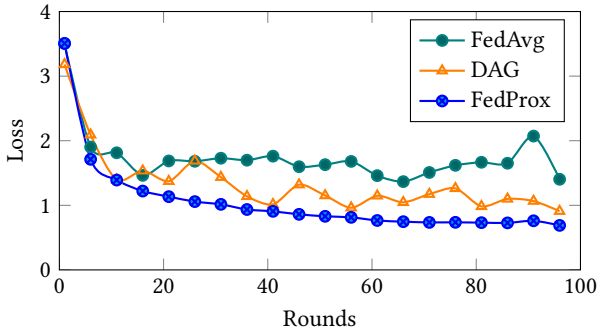


Figure 11: The DAG consistently performs better with regards to the average loss compared to FedAvg. A small margin remains compared to the centralized FedProx approach.

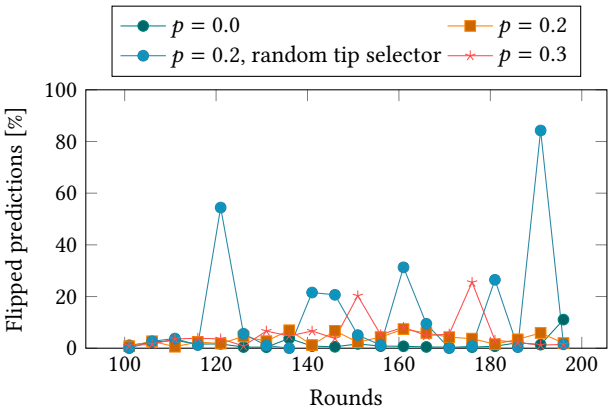


Figure 12: Flipped predictions of samples in the classes 3 and 8.

of the handwritten digits. Specifically, we exchanged the labels 3 and 8 for a subset of clients after 100 training rounds without any data poisoning.

Figure 12 illustrates the success of the poisoning attack in different scenarios: It shows how many samples of the classes 3 and 8 in the clients' test datasets were mispredicted as belonging to the other class using the reference model that the clients selected from the DAG. The parameter p defines the fraction of clients that used poisoned training and test data. As a baseline, we also measured the behavior of the original, purely random tip selector.

Compared to the baseline results, the effects of the attack with $p = 0.2$ on the overall network are very limited and almost within the variance that is also present with $p = 0.0$. When increasing the number of poisoned clients to $p = 0.3$, the effects are noticeable, but still below 30% mispredictions overall.

Looking at the selected reference transactions in more detail, we can observe that, although the poisoning did not have severe effects, a high number of poisoned updates are included in the reference transactions by direct or indirect approvals. Especially noteworthy is that the poisoning impact on the mispredictions is higher for the random tip selector with $p = 0.2$ than for the

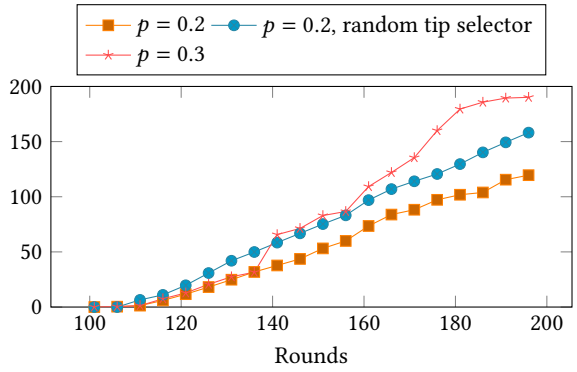


Figure 13: Average number of approved poisonous transactions in the consensus.

accuracy tip selector with $p = 0.3$, even though the number of approved poisoned transactions is lower for the former.

This can be explained by the containment of poisoned transactions within a subset of clients. While a poisoned transaction may be incorporated into another clusters' consensus from time to time, leading to a high number of indirectly approved poisoned transactions, in most cases it is other malicious clients that approve a poisoned transaction:

Figure 14 depicts the distribution of poisoned clients over the clusters reconstructed by the Louvain algorithm. Most of them end up in clusters where a majority of other clients are also affected by the attack.

While this protects other network participants, it also means that the attack is difficult to detect for the affected clients. If the goal of the attack is known in advance, clients could use the random tip selector to obtain a reference transaction that is most likely not affected by the attack in order to cross-check their locally trained model.

5.3.5 Scalability. Compared to gossiping approaches for federated learning, the main overhead of the proposed DAG consists of the time that each client needs to perform the random walk and to evaluate models on the local data as part of it. If the time needed for the random walk increases with the number of updates in the DAG or the number of participating clients, this would limit the scalability of the approach. This section evaluates the time needed

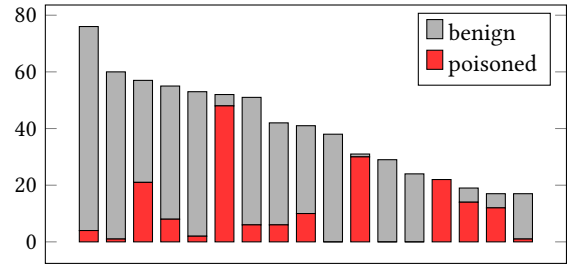


Figure 14: Distribution of poisoned clients over 15 inferred clusters for $p = 0.3$.

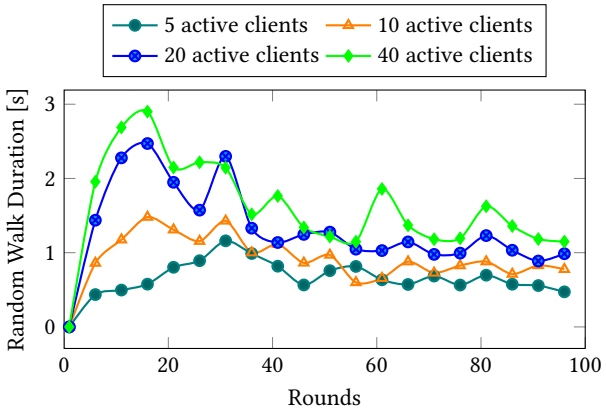


Figure 15: Development of time required for the random walk for different numbers of concurrently active clients over the course of 100 training rounds. The differences for increasing numbers of active clients are marginal which indicates a good scalability of the approach.

for the random walk when training the original FMNIST dataset with increasing numbers of clients that are concurrently performing model training.

Generally, the random walk makes up a significant amount of the required compute resources compared to model training: In our example, training the FMNIST model takes about 300ms whereas the time required for the random walk ranges from 600-1200ms (cf. Figure 15). However, in a real-world implementation, the time required for the random walk can be hidden between training runs, since it would be sensible for a client to only perform training in set intervals or when new training data arrives.

In our scalability experiments, we started the random walk at a transaction sampled at a depth of 15-25 transactions from the tips, as proposed by Popov [58].

Figure 15 shows the average time that a single client spends on the random walk, for increasing numbers of clients that are training concurrently. The concurrency in the network has an impact on the random walk because well- performing models will generally have a greater number of direct child transactions that were created simultaneously and all of which have to be evaluated on local data during the random walk.

Especially in the early phases of the collaborative training process, this number is not well balanced among the transactions since there are still large differences in accuracy between them. However, as the trained models improve, the variance in the number of child transaction levels out. In practice, this would also require ideal network conditions, i.e. all new transactions are broadcasted equally well among network participants.

In conclusion, the concurrency in the network has little impact on the costs incurred by the DAG algorithm, and hence it can be expected to scale well also for larger numbers of clients.

6 CONCLUSION AND FUTURE WORK

We presented a novel approach to achieve specialized models in federated learning without a central server: by using a DAG for the

communication of models and an accuracy-biased random walk, we show the manifestation of clusters of clients with similar local data. This specialization emerges directly from the way the DAG is used to communicate model updates, thus creating a unified solution for decentralized and personalized federated learning. We enable a tradeoff between reaching a consensus on a generalized model and specializing (personalizing) the models to local data. We evaluated this approach with our prototypical implementation on three datasets and showed equal or better learning performance in a simulation. Finally, we showed the poisoning resistance and scalability of our approach.

In the future we would like to integrate ideas from multi-task and personalized federated learning such as training only some layers of the machine learning model.

ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers for their helpful comments on earlier versions of this paper. We are also thankful for the comments by Felix Eberhardt and Daniel Richter from our research group.

This research was partly funded by the Federal Ministry for Economic Affairs and Energy of Germany as part of the program "Smart Data" (project number 01MD19014C), by the German Federal Ministry of Transport and Digital Infrastructure through the mFUND (project number 19F2093C) and by the Federal Ministry of Education and Research of Germany in the framework of KI-LAB-ITSE (project number 01IS19066).

REFERENCES

[1] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. 2008. Fast Unfolding of Communities in Large Networks. *Journal of Statistical Mechanics: Theory and Experiment* (Oct. 2008). https://doi.org/10.1088/1742-5468/2008/10/P10008

[2] Michael Blot, David Picard, Nicolas Thome, and Matthieu Cord. 2019. Distributed optimization for deep learning with gossip exchange. *Neurocomputing* 330 (2019), 287–296. https://doi.org/10.1016/j.neucom.2018.11.002

[3] Vitalik Buterin. 2014. A Next-Generation Smart Contract and Decentralized Application Platform. *white paper* (2014).

[4] Vitalik Buterin and Virgil Griffith. 2017. Casper the Friendly Finality Gadget. *arXiv preprint arXiv:1710.09437* (2017).

[5] Sebastian Caldas, Sai Meher Karthik Duddu, Peter Wu, Tian Li, Jakub Konečný, H. Brendan McMahan, Virginia Smith, and Ameet Talwalkar. 2019. LEAF: A Benchmark for Federated Settings. *arXiv:1812.01097 [cs, stat]* (Dec. 2019).

[6] Rich Caruana. 1997. Multitask Learning. *Machine Learning* (July 1997). https://doi.org/10.1023/A:1007379606734

[7] Haoye Chai, Supeng Leng, Yijin Chen, and Ke Zhang. 2020. A Hierarchical Blockchain-Enabled Federated Learning Algorithm for Knowledge Sharing in Internet of Vehicles. *IEEE Transactions on Intelligent Transportation Systems* (2020). https://doi.org/10.1109/TITS.2020.3002712

[8] Xuhui Chen, Jinlong Ji, Changqing Luo, Weixian Liao, and Pan Li. 2018. When Machine Learning Meets Blockchain: A Decentralized, Privacy-Preserving and Secure Design. In *2018 IEEE International Conference on Big Data (Big Data)*. IEEE. https://doi.org/10.1109/bigdata.2018.8622598

[9] Yujing Chen, Yue Ning, Martin Slawski, and Huzefa Rangwala. 2020. Asynchronous Online Federated Learning for Edge Devices with Non-IID Data. *arXiv:1911.02134 [cs]* (Oct. 2020).

[10] Anton Churyumov. 2016. Byteball: A Decentralized System for Storage and Transfer of Value. (2016). https://byteball.org/Byteball.pdf

[11] Luca Corinzia and Joachim M. Buhmann. 2019. Variational Federated Multi-Task Learning. *arXiv:1906.06268 [cs, stat]* (June 2019).

[12] Kyle Croman, Christian Decker, Ittay Eyal, Adem Efe Gencer, Ari Juels, Ahmed Kosba, Andrew Miller, Prateek Saxena, Elaine Shi, and Emin Gün Sirer. 2016. On Scaling Decentralized Blockchains. In *International Conference on Financial Cryptography and Data Security*. Springer.

[13] Gábor Danner and Márk Jelasity. 2018. Token Account Algorithms: The Best of the Proactive and Reactive Worlds. In *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*. 885–895. https://doi.org/10.1109/ICDCS.2018.00090

[14] Yuyang Deng, Mohammad Mahdi Kamani, and Mehrdad Mahdavi. 2020. Adaptive Personalized Federated Learning. *arXiv:2003.13461 [cs, stat]* (Nov. 2020).

[15] Mina Aghaei Dinani, Adrian Holzer, Hung Nguyen, Marco Ajmone Marsan, and Gianluca Rizzo. 2021. Gossip Learning of Personalized Models for Vehicle Trajectory Prediction. In *2021 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*. 1–7. https://doi.org/10.1109/WCNCW49093.2021.9420038

[16] Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. 2020. Personalized Federated Learning: A Meta-Learning Approach. *arXiv:2002.07948 [cs, math, stat]* (Oct. 2020).

[17] Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. *arXiv:1703.03400 [cs]* (July 2017).

[18] Lodovico Giaretta and Šarūnas Girdzijauskas. 2019. Gossip Learning: Off the Beaten Path. In *2019 IEEE International Conference on Big Data (Big Data)*. 1117–1124. https://doi.org/10.1109/BigData47090.2019.9006216

[19] Yossi Gilad, Rotem Hemo, Silvio Micali, Georgios Vlachos, and Nickolai Zeldovich. 2017. Algorand: Scaling Byzantine Agreements for Cryptocurrencies. In *Proceedings of the 26th Symposium on Operating Systems Principles*.

[20] Project Gutenberg. 2020. Project Gutenberg. https://www.gutenberg.org/.

[21] Corentin Hardy, Erwan Le Merrer, and Bruno Sericola. 2018. Gossiping GANs: Position Paper. In *Proceedings of the Second Workshop on Distributed Infrastructures for Deep Learning* (Rennes, France) *(DIDL '18)*. Association for Computing Machinery, New York, NY, USA, 25–28. https://doi.org/10.1145/3286490.3286563

[22] István Hegedűs, Gábor Danner, and Márk Jelasity. 2019. Gossip Learning as a Decentralized Alternative to Federated Learning. In *Distributed Applications and Interoperable Systems*, José Pereira and Laura Ricci (Eds.). Springer International Publishing, Cham, 74–90.

[23] István Hegedűs, Gábor Danner, and Márk Jelasity. 2021. Decentralized learning works: An empirical comparison of gossip learning and federated learning. *J. Parallel and Distrib. Comput.* 148 (2021), 109–124. https://doi.org/10.1016/j.jpdc.2020.10.006

[24] Kevin Hsieh, Amar Phanishayee, Onur Mutlu, and Phillip B. Gibbons. 2020. The Non-IID Data Quagmire of Decentralized Machine Learning. *arXiv:1910.00189 [cs, stat]* (Aug. 2020).

[25] Chenghao Hu, Jingyan Jiang, and Zhi Wang. 2019. Decentralized Federated Learning: A Segmented Gossip Approach. *CoRR* abs/1908.07782 (2019). arXiv:1908.07782 http://arxiv.org/abs/1908.07782

[26] G. Hua, L. Zhu, J. Wu, C. Shen, L. Zhou, and Q. Lin. 2020. Blockchain-Based Federated Learning for Intelligent Control in Heavy Haul Railway. *IEEE Access* (2020). https://doi.org/10.1109/ACCESS.2020.3021253

[27] Gur Huberman, Jacob Leshno, and Ciamac C. Moallemi. 2017. Monopoly without a Monopolist: An Economic Analysis of the Bitcoin Payment System. *Bank of Finland Research Discussion Paper* (2017).

[28] Laurent Jacob, Jean-philippe Vert, and Francis Bach. 2008. Clustered Multi-Task Learning: A Convex Formulation. *Advances in Neural Information Processing Systems* (2008).

[29] Yihan Jiang, Jakub Konečný, Keith Rush, and Sreeram Kannan. 2019. Improving Federated Learning Personalization via Model Agnostic Meta Learning. *arXiv:1909.12488 [cs, stat]* (Sept. 2019).

[30] Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Keith Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, Rafael G. L. D'Oliveira, Salim El Rouayheb, David Evans, Josh Gardner, Zachary Garrett, Adrià Gascón, Badih Ghazi, Phillip B. Gibbons, Marco Gruteser, Zaid Harchaoui, Chaoyang He, Lie He, Zhouyuan Huo, Ben Hutchinson, Justin Hsu, Martin Jaggi, Tara Javidi, Gauri Joshi, Mikhail Khodak, Jakub Konečný, Aleksandra Korolova, Farinaz Koushanfar, Sanmi Koyejo, Tancrède Lepoint, Yang Liu, Prateek Mittal, Mehryar Mohri, Richard Nock, Ayfer Özgür, Rasmus Pagh, Mariana Raykova, Hang Qi, Daniel Ramage, Ramesh Raskar, Dawn Song, Weikang Song, Sebastian U. Stich, Ziteng Sun, Ananda Theertha Suresh, Florian Tramèr, Praneeth Vepakomma, Jianyu Wang, Li Xiong, Zheng Xu, Qiang Yang, Felix X. Yu, Han Yu, and Sen Zhao. 2019. Advances and Open Problems in Federated Learning. *arXiv:1912.04977 [cs, stat]* (Dec. 2019).

[31] Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. 2017. Ouroboros: A Provably Secure Proof-of-Stake Blockchain Protocol. In *Annual International Cryptology Conference*. Springer.

[32] H. Kim, J. Park, M. Bennis, and S. Kim. 2019. Blockchained On-Device Federated Learning. *IEEE Communications Letters* (2019). https://doi.org/10.1109/LCOMM.2019.2921755

[33] Y. J. Kim and C. S. Hong. 2019. Blockchain-Based Node-Aware Dynamic Weighting Methods for Improving Federated Learning Performance. In *2019 20th Asia-Pacific Network Operations and Management Symposium (APNOMS)*. https://doi.org/10.23919/APNOMS.2019.8893114

[34] Alex Krizhevsky. 2009. *Learning multiple layers of features from tiny images*. Technical Report.

[35] Rajesh Kumar, Abdullah Aman Khan, Sinmin Zhang, Jay Kumar, Ting Yang, Noorbakhash Amiri Golalirz, Zakria, Ikram Ali, Sidra Shafiq, and WenYong Wang. 2020. Blockchain-Federated-Learning and Deep Learning Models for COVID-19 Detection Using CT Imaging. *arXiv:2007.06537 [cs, eess]* (Dec. 2020).
[36] Yann LeCun, Corinna Cortes, and CJ Burges. 2010. MNIST handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist* 2 (2010).
[37] Chenxing Li, Peilun Li, Dong Zhou, Wei Xu, Fan Long, and Andrew Yao. 2018. Scaling Nakamoto Consensus to Thousands of Transactions per Second. *arXiv:1805.03870 [cs]* (Aug. 2018).
[38] Tian Li, Shengyuan Hu, Ahmad Beirami, and Virginia Smith. [n.d.]. *Ditto: Fair and Robust Federated Learning Through Personalization.* arXiv:2012.04221 [cs, stat] http://arxiv.org/abs/2012.04221
[39] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. [n.d.]. *Federated Optimization in Heterogeneous Networks.* arXiv:1812.06127 [cs, stat] http://arxiv.org/abs/1812.06127
[40] Wei Li and Andrew McCallum. 2006. Pachinko Allocation: DAG-Structured Mixture Models of Topic Correlations. In *Proceedings of the 23rd International Conference on Machine Learning (ICML '06)*. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/1143844.1143917
[41] Yunlong Lu, Xiaohong Huang, Yueyue Dai, Sabita Maharjan, and Yan Zhang. 2020. Differentially Private Asynchronous Federated Learning for Mobile Edge Computing in Urban Informatics. *IEEE Transactions on Industrial Informatics* (March 2020). https://doi.org/10.1109/TII.2019.2942179
[42] Loi Luu, Viswesh Narayanan, Chaodong Zheng, Kunal Baweja, Seth Gilbert, and Prateek Saxena. 2016. A Secure Sharding Protocol for Open Blockchains. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*.
[43] Lingjuan Lyu, Jiangshan Yu, Karthik Nandakumar, Yitong Li, Xingjun Ma, Jiong Jin, Han Yu, and Kee Siong Ng. 2020. Towards Fair and Privacy-Preserving Federated Deep Models. *IEEE Transactions on Parallel and Distributed Systems* (Nov. 2020). https://doi.org/10.1109/TPDS.2020.2996273
[44] Yishay Mansour, Mehryar Mohri, Jae Ro, and Ananda Theertha Suresh. 2020. Three Approaches for Personalization with Applications to Federated Learning. *arXiv:2002.10619 [cs, stat]* (July 2020).
[45] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *Artificial Intelligence and Statistics*. PMLR.
[46] Satoshi Nakamoto. 2008. Bitcoin: A Peer-to-Peer Electronic Cash System. (2008).
[47] M. E. J. Newman. 2006. Modularity and Community Structure in Networks. *Proceedings of the National Academy of Sciences* (June 2006). https://doi.org/10.1073/pnas.0601602103
[48] M. E. J. Newman and M. Girvan. 2004. Finding and Evaluating Community Structure in Networks. *Physical Review E* (Feb. 2004). https://doi.org/10.1103/PhysRevE.69.026113
[49] Minh N. H. Nguyen, Shashi Raj Pandey, Tri Nguyen Dang, Eui-Nam Huh, Choong Seon Hong, Nguyen H. Tran, and Walid Saad. 2020. Self-Organizing Democratized Learning: Towards Large-Scale Distributed Learning Systems. *arXiv:2007.03278 [cs, stat]* (July 2020).
[50] Minh N. H. Nguyen, Shashi Raj Pandey, Kyi Thar, Nguyen H. Tran, Mingzhe Chen, Walid Saad, and Choong Seon Hong. 2020. Distributed and Democratized Learning: Philosophy and Research Challenges. *arXiv:2003.09301 [cs, stat]* (Oct. 2020).
[51] Róbert Ormándi, István Hegedűs, and Márk Jelasity. 2013. Gossip learning with linear models on fully distributed data. *Concurrency and Computation: Practice and Experience* 25, 4 (2013), 556–571. https://doi.org/10.1002/cpe.2858 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/cpe.2858
[52] Joseph Poon and Thaddeus Dryja. 2016. *The Bitcoin Lightning Network: Scalable off-Chain Instant Payments.*
[53] Youyang Qu, Longxiang Gao, Tom H. Luan, Yong Xiang, Shui Yu, Bai Li, and Gavin Zheng. 2020. Decentralized Privacy Using Blockchain-Enabled Federated Learning in Fog Computing. *IEEE Internet of Things Journal* (June 2020). https://doi.org/10.1109/JIOT.2020.2977383
[54] M. A. Rahman, M. S. Hossain, M. S. Islam, N. A. Alrajeh, and G. Muhammad. 2020. Secure and Provenance Enhanced Internet of Health Things Framework: A Blockchain Managed Federated Learning Approach. *IEEE Access* (2020). https://doi.org/10.1109/ACCESS.2020.3037474
[55] Team Rocket. 2018. Snowflake to Avalanche : A Novel Metastable Consensus Protocol Family for Cryptocurrencies. /paper/Snowflake-to-Avalanche-%3A-A-Novel-Metastable-Family/85ec19594046bbcfe12137c7c2e3744677129820.
[56] Sebastian Ruder. 2017. An Overview of Multi-Task Learning in Deep Neural Networks. *arXiv:1706.05098 [cs, stat]* (June 2017).
[57] Robert Schmid, Bjarne Pfitzner, Jossekin Beilharz, Bert Arnrich, and Andreas Polze. 2020. Tangle Ledger for Decentralized Learning. In *2020 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*. https://doi.org/10.1109/IPDPSW50202.2020.00144
[58] Serguei Popov. 2017. *The Tangle.*
[59] Muhammad Shayan, Clement Fung, Chris JM Yoon, and Ivan Beschastnikh. 2018. Biscotti: A Ledger for Private and Secure Peer-to-Peer Machine Learning. *arXiv preprint arXiv:1811.09904* (2018).
[60] Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet S. Talwalkar. 2017. Federated Multi-Task Learning. *Advances in Neural Information Processing Systems* (2017).
[61] Yonatan Sompolinsky, Yoad Lewenberg, and Aviv Zohar. 2016. SPECTRE: A Fast and Scalable Cryptocurrency Protocol. *IACR Cryptology ePrint Archive* (2016).
[62] Richard Tomsett, Kevin Chan, and Supriyo Chakraborty. 2019. Model poisoning attacks against distributed machine learning systems. In *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications*, Tien Pham (Ed.), Vol. 11006. International Society for Optics and Photonics, SPIE, 481 – 489. https://doi.org/10.1117/12.2520275
[63] Kentaroh Toyoda and Allan N. Zhang. 2019. Mechanism Design for An Incentive-Aware Blockchain-Enabled Federated Learning Platform. In *2019 IEEE International Conference on Big Data (Big Data)*. IEEE, Los Angeles, CA, USA. https://doi.org/10.1109/BigData47090.2019.9006344
[64] Paul Vanhaesebrouck, Aurélien Bellet, and Marc Tommasi. 2017. Decentralized Collaborative Learning of Personalized Models over Networks. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (Proceedings of Machine Learning Research, Vol. 54)*, Aarti Singh and Jerry Zhu (Eds.). PMLR, 509–517. https://proceedings.mlr.press/v54/vanhaesebrouck17a.html
[65] Kangkang Wang, Rajiv Mathews, Chloé Kiddon, Hubert Eichner, Françoise Beaufays, and Daniel Ramage. 2019. Federated Evaluation of On-Device Personalization. *arXiv:1910.10252 [cs, stat]* (Oct. 2019).
[66] Zhibo Wang, Mengkai Song, Zhifei Zhang, Yang Song, Qian Wang, and Hairong Qi. 2018. Beyond Inferring Class Representatives: User-Level Privacy Leakage from Federated Learning. *CoRR* (2018).
[67] J. Weng, J. Weng, J. Zhang, M. Li, Y. Zhang, and W. Luo. 2019. DeepChain: Auditable and Privacy-Preserving Deep Learning with Blockchain-Based Incentive. *IEEE Transactions on Dependable and Secure Computing* (2019). https://doi.org/10.1109/TDSC.2019.2952332
[68] Yu Zhang and Dit-Yan Yeung. 2010. A Convex Formulation for Learning Task Relationships in Multi-Task Learning. In *Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence* (Catalina Island, CA) *(UAI'10)*. AUAI Press, Arlington, Virginia, USA, 10 pages.
[69] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. 2018. Federated Learning with Non-IID Data. *arXiv:1806.00582 [cs, stat]* (June 2018).
[70] Yang Zhao, Jun Zhao, Linshan Jiang, Rui Tan, Dusit Niyato, Zengxiang Li, Lingjuan Lyu, and Yingbo Liu. 2020. Privacy-Preserving Blockchain-Based Federated Learning for IoT Devices. *IEEE Internet of Things Journal* (Aug. 2020). https://doi.org/10.1109/JIOT.2020.3017377
[71] Jiayu Zhou, Jianhui Chen, and Jieping Ye. 2011. Clustered Multi-Task Learning Via Alternating Structure Optimization. In *Advances in Neural Information Processing Systems*.