Towards Sybil Resilience in Decentralized Learning

— MSc. Thesis —

Thomas Werthenbach
Delft University of Technology
Delft, The Netherlands
T.A.K.Werthenbach@student.tudelft.nl

Johan Pouwelse
Delft University of Technology
Delft, The Netherlands
J.A.Pouwelse@tudelft.nl

***Abstract*—**

***Index Terms*—decentralized learning, sybil attack, sybil resilience**

I. Introduction

The rise of machine learning has resulted in an increasing number of everyday-life intelligent applications. As such, machine learning has been used in personal assistants [1], recommendation in social media [2] and music [3], and cybersecurity [4]. However, accurate machine learning models require large training datasets [5], [6], which can often be hard to obtain and store due to recent privacy legislation [7]. Federated learning [8] has become a promising alternative and widely adopted tool for crowd sourcing computationally expensive machine learning operations, reportedly having been used for training numerous industrial machine learning models [9]–[13]. Federated learning ensures the protection of privacy, as the user's data will not leave their device during training.

With federated learning, in contrast to centralized machine learning, training takes place on the end-users' personal devices, which are often referred to *edge devices* or *nodes*. The resulting trained models are communicated to a central server, commonly referred to as the *parameter server*, which aggregates these models using some predefined methodology. By only sharing the end user-trained models with the parameter server, the user's privacy is preserved, while obtaining comparable performance compared to centralized machine learning [14]. While there exist attacks in which training data can be reconstructed based on the gradient of the trained models [15], [16], defense mechanisms against this attack have been proposed [17], [18].

However, federated learning suffers from some disadvantages. For instance, the parameter server aggregates the models of all participating nodes, inducing heavy communication costs and a potential bottleneck in the learning process affecting the overall convergence time [19]. Secondly, the scalability in terms of the amount of nodes heavily varies depending on the aggregation method. In secure and robust federated learning aggregation methods, the incorporation of additional nodes during aggregation may result in significantly increased computational effort for the parameter server [20]. Thirdly, the parameter server performing the aggregation poses a single-point of failure [21]. Disruptions to the parameter server can cause downtime and hinder the overall model training process, particularly in architectures where nodes require the globally aggregated model before continuing their training. An upcoming alternative aiming to resolve these issues is *decentralized learning*, also commonly referred to as *decentralized federated learning*. In decentralized learning, there exists no dedicated parameter server performing the aggregation and the nodes form a distributed network, e.g. a peer-to-peer network, in which each node individually performs aggregation on their neighbours' models (see Figure 1). While the information available during aggregation is more limited relative to federated learning, it has been shown that decentralized learning has the potential to obtain similar results compared to federated learning [22]. Models are exchanged between individual devices and aggregated on individual scale using some predefined aggregation method, alleviating the communicative bottleneck and single point of failure issues imposed on federated learning, and paving the path for boundless scalability.

While decentralized learning solves the scalability challenges faced in federated learning, it is still vulnerable to byzantine environments [23]. Since the predefined aggregation method in decentralized learning does not have access to all models in the network, aggregation is performed with less information compared to federated learning, resulting in relatively less resistance against possible poisoning attacks [24]. Poisoning attacks are can generally be categorized in two categories, namely those of *targeted poisoning attacks* and *untargeted poisoning attacks*. Targeted poisoning attacks focus on achieving a specific goal an adversary aims to achieve such as the label-flipping attack [25], [26] and the backdoor attack [27]–[29]. On the other hand, untargeted poisoning attacks aim to hinder the result of the training process in some way without any particular goal in mind. The effect of these attacks can often be amplified through combining them with the Sybil attack [30], in which an adversary controls a substantial amount of nodes to increase its influence. As such, an adversary may deploy the Sybil attack to rapidly spread their poisoned model through the network. In this work, we focus exclusively on targeted poisoning attacks amplified by Sybil attacks in decentralized learning.

Prior work on resilience against poisoning attacks combined with Sybil attacks in distributed machine learning has mainly

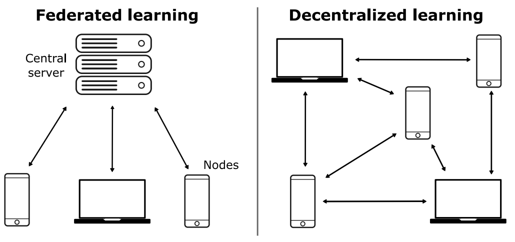


Figure 1: Federated learning compared to decentralized learning. Arrows represent a connection between two nodes and indicates the two connecting nodes share model updates during each training round.

been done in federated learning settings. One popular example of such work is *FoolsGold* [31], which aims to increase Sybil resilience under the assumption that all Sybils will broadcast similar gradients during each round of training. By dynamically adapting the aggregation weights of peers' models based on their similarity with others, experimental results suggest that FoolsGold has the potential to provide effective protection against Sybil attacks in small-scale and simple federated learning settings.

In this work, we experimentally demonstrate *FoolsGold*'s inability to scale to an unbounded number of nodes in federated learning and inept defensive capabilities against targeted poisoning attacks in decentralized learning. We suggest an improved version of FoolsGold, named Sydle[1], which shows significant resilience towards defending against targeted poisoning attacks whilst enjoying the boundless scalability offered by decentralized learning. More specifically, we achieve this by introducing a probabilistic gossiping mechanism for knowledge spreading. Finally, we empirically evaluate this algorithm on numerous types of Sybil attacks and show its ability to obtain increased Sybil resilience.

To the best of our knowledge, there exists only a single other work on defensive algorithms against poisoning attack in decentralized learning [32]. Moreover, this paper is the first to study Sybil attacks in decentralized learning. In short, our contributions are the following:

- We evaluate FoolsGold, a popular Sybil resilience algorithm in federated learning, and assess its compatibility with decentralized learning in Section III.
- We present Sydle, a pioneering algorithm for Sybil resilience with boundless scalability in decentralized learning, in Section V.
- We perform an empirical evaluation of Sydle's performance in VI.
- Maybe: We provide a convergence analysis on Sydle in section VII.

II. Background

A. Federated learning

Federated learning was initially proposed by Google [8] as a means for training machine learning models on real user data without compromising users' privacy. This is achieved by training the machine learning model on the edge devices, which contain the real user data. The training proceeds in synchronous rounds, each consisting of a predefined number of epochs, during which the trained models are sent to the parameter server at the end of each round. The role of the parameter server is to aggregate all trained models into a global model without the need of any training data. After the models are aggregated, the global model is communicated back to the edge devices, after which the next training round commences. See Figure 1 for a simplified federated learning network topology. The original paper [8] suggests the usage of FedAvg, which adopts a weighted average function as the aggregation function, such that the next global model w^{t+1} is calculated as follows:

$$w^{t+1} = \sum_{i \in N} \frac{|\mathcal{D}_i|}{|\mathcal{D}|} w_i^t \tag{1}$$

where w_i^t is the model of node i in round t, N is the set of nodes, $\mathcal{D}_i$ corresponds to node i's local dataset and $\mathcal{D}$ is the global distributed collection of data, such that $\mathcal{D} = \bigcup_{j \in S} \mathcal{D}_j$.

The goal is of the training process is to minimize the global loss function such that the global model x approaches the optimal model x^*. More formally, the search for a global optimal model can approximately be defined as:

$$w^* = \arg\min_w \sum_{i \in N} \frac{|\mathcal{D}_i|}{|\mathcal{D}|} \mathcal{L}_i(w) \tag{2}$$

where $\mathcal{L}_i$ is a node's loss function, e.g. cross-entropy loss or negative log likelihood loss, using the node i's local dataset.

[1]Sydle stands for SYbil resilient Decentralized LEarning.

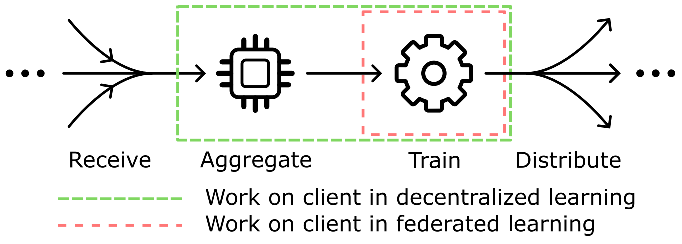

Receive
Aggregate
Train
Distribute
Work on client in decentralized learning
Work on client in federated learning

Figure 2: Todo: caption.

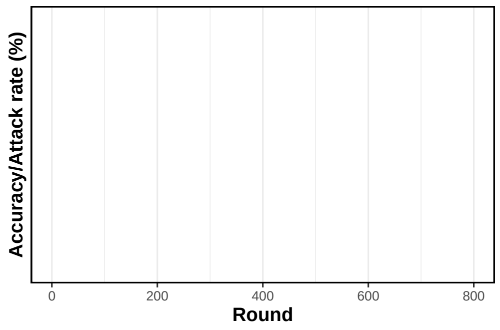


Figure 3: Sydle compared to a number of other algorithms.

B. Decentralized learning

Decentralized learning is an upcoming alternative for federated learning [33]–[36]. In contrast to federated learning, which relies on a parameter server for aggregating locally trained models, aggregation in decentralized learning takes place at a smaller scale and is performed by every participating node on their own model and those of its neighbours (see Figure 1) By doing so, numerous issues faced in federated learning can be resolved; the most notable improvement of which can be found in terms of scalability and can be decomposed into three distinct aspects:

1) *Communication*: In parameter server-centered aggregation techniques, all participating models are downloaded and uploaded every training round, forming a communication bottleneck bounded by the parameter server's internet connection. Such bottlenecks are reduced in decentralized learning to the number of neighbours.
2) *Memory*: Storing all models in memory during aggregation, may result in substantial memory usage. In decentralized learning, this constraint poses a diminished issue, given that the aggregation transpires with a significantly reduced number of models.
3) *Aggregation*: The aggregation function is no longer performed with every model from all participating nodes, reducing the required computation time of more sophisticated aggregation functions, particularly those that do not scale linearly with respect to the number of models.

If the prior example of a basic averaging function were to be applied in decentralized learning, one would obtain the following aggregation function:

$$w_i^{t+1} = \frac{1}{|\mathcal{N}_i| + 1} \sum_{j \in \{\mathcal{N}_i \,\cup\, i\}} w_j^t \tag{3}$$

where $\mathcal{N}_i$ is the set of all neighbours of node i.

Nodes in decentralized learning may not have access to all information in the network, causing a decrease in informativeness. More specifically, convergence speeds may decrease compared to federated learning [36] and nodes may experience increased vulnerability to byzantine attacks due to the lack of global information [23]. However, it has been shown that decentralized learning can obtain comparable performance results relative to federated learning and, in some cases, outperform federated learning altogether [37], [38]. Furthermore, multiple defensive strategies focused on decentralized learning have been proposed recently [31], [32], [39]–[41].

C. Targeted poisoning attacks

Poisoning attacks can be defined as methods in which an adversary attempts to compromise the integrity of the global model and can be taxonomized into two categories: targeted poisoning attacks and untargeted poisoning attacks. With untargeted poisoning attacks, the adversary aims to decrease the performance metric of the model without any particular goal in mind. On the other hand, in the context of targeted poisoning attacks, the adversary aims to achieve a predetermined goal by manipulating the global model to behave in a certain deterministic manner which deviates from the objectively correct behaviour. We consider two of such attacks related to the domain of classification: the label-flipping attack [25], [26] and the backdoor attack [27]–[29].

The label-flipping attack can be deployed as an attempt to increase the likelihood for two targeted classes to be misclassified. More specifically, given two target classes t_1 and t_2, the aim of the label-flipping attack is to manipulate the model such that some arbitrary sample $x \in X_{t_1}$ belonging to class t_1 is more likely to be classified as class t_2 by the global model and vice versa. A way to achieve this is to explicitly transform the adversary's local dataset $\mathcal{D}$ to an adversarial dataset $\mathcal{D}'$ and train the adversarial model on this dataset. Given two target classes t_1 and t_2, this transformation can be defined as:

$$\begin{aligned} \mathcal{D}' = {} & \{(x, y) \in \mathcal{D} \mid y \neq t_1 \wedge y \neq t_2\} \\ & \cup \{(x, t_1) \mid (x, y) \in \mathcal{D},\ y = t_2\} \\ & \cup \{(x, t_2) \mid (x, y) \in \mathcal{D},\ y = t_1\} \end{aligned} \tag{4}$$

The backdoor attack requires a more extensive manipulation of the training data. The objective of a backdoor attack is to alter the global model such that any sample containing a specific predefined pattern is misclassified to a chosen target class. In the domain of image classification, this adversarial pattern could for instance correspond to small square or triangle in the top-left corner of the input image. Given a target class t and a function f to introduce a hidden pattern to input samples, the dataset transformation applied on the adversary's local dataset $\mathcal{D}$ can be defined as:

$$\mathcal{D}' = \{(f(x), t) \mid (x, y) \in \mathcal{D}\} \tag{5}$$

D. The Sybil attack

The Sybil attack, first introduced by Douceur [30], is an adversarial strategy in decentralized environments in which the attacker exploits the inability of honest nodes to verify the authenticity of another node's identity. By effortlessly creating fake nodes, named *Sybils*, and strategically connecting these to nodes in the targeted decentralized network, the attacker may gain significantly more influence compared to the honest nodes. We denote the connections between Sybils and honest nodes as *attack edges*. A typical example of a scenario in which the Sybil attack may be deployed is *voting* [42], [43]. In such a case, an attacker can easily generate sufficient nodes to outnumber all other real voters.

In this work, we consider the Sybil poisoning attack, in which an attacker creates fake nodes to spread its malicious model more rapidly and effectively throughout the network.

III. Related work

A. FoolsGold

FoolsGold [31] is an algorithm for mitigating Sybil poisoning attacks in federated learning settings. It builds on the assumption that Sybil model updates show a substantially higher degree of similarity relative to that of honest model updates. Through the computation of similarity between a node's model update history and that of others, and subsequently mapping this to the model update's weight in an average-based aggregation, FoolsGold manages to mitigate Sybil targeted poisoning attacks as shown in Figure 4.

During aggregation, FoolsGold first computes the pairwise cosine similarity score of all model update histories. The model update history of node i in round T is defined as $h_i^T = \sum_{t=0}^{T} w_i^t$. In an effort to decrease the number of false positives among honest nodes, each score s_{ij} is multiplied by the ratio of node i's maximum score and node j's maximum score in cases where the latter is greater, such that $\frac{\max_v s_{iv}}{\max_v s_{jv}}$ if $\max_v s_{iv} < \max_v s_{jv}$. The scores are then aggregated per node by taking the maximum and subsequently inversed, such that node i's aggregated score s_i' can be defined as $s_i' = 1 - \max_v s_{iv}$. As FoolsGold assumes there exists at least one honest node, the aggregated scores are rescaled such that the maximum aggregated score equals 1.

The aggregated scores are then transformed to weights for average-based aggregation through the use of a bounded logit function. This function can be considered a gradual decision boundary for determining a node's honesty based on its similarity with others. Finally, the weights are normalized and the aggregated model is computed through weighted average.

A reproduction of FoolsGold's results can be found in Figure 4, where the attack rate represents the extent to which the attack was successful, e.g. the percentage of labels that are successfully flipped in the label-flipping attack. It becomes clear that FoolsGold shows significantly higher Sybil resilience compared to the FedAvg. However, as discussed in Section II-B, federated learning can be considered unscalable as the number of participating nodes increases, particularly in view of the $\mathcal{O}(n^2)$ pairwise cosine similarity computation required by FoolsGold. Figure 5 shows the performance of FoolsGold in a decentralized setting against the performance of our improved algorithm, Sydle, based upon FoolsGold. It is clear that FoolsGold's performance can heavily depend based on the network topology, whereas Sydle shows comparatively more Sybil resilience.

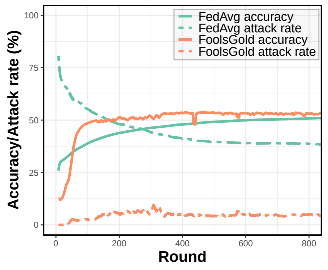


Figure 4: FoolsGold and FedAvg in federated learning setting using the CIFAR-10 [44] dataset on a LeNet-5 [45] model.

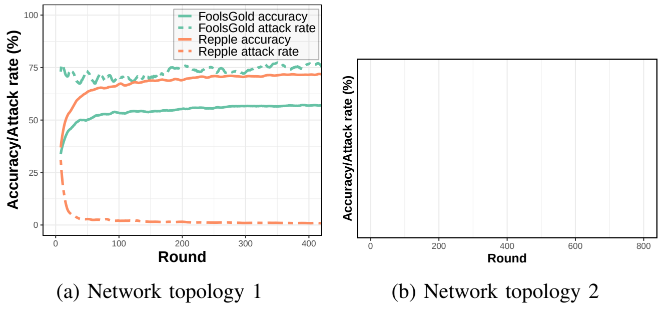


Figure 5: FoolsGold and Sydle in decentralized learning using the FashionMNIST [46] dataset on a single-layer softmax neural network.

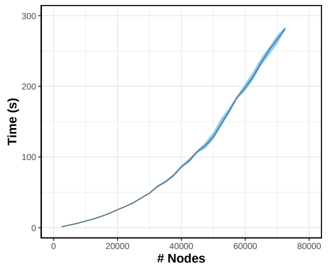


Figure 6: Pairwise cosine similarity computation time against the number of nodes, illustrating the $\mathcal{O}(n^2)$ time complexity of the pairwise cosine similarity computation.

B. Resilient Averaging Gradient Descent

TODO Resilient Averaging Gradient Descent (RAGD) [32] is a novel algorithm for mitigating poisoning attacks in decentralized learning.
How our work is different:

- RAGD naively assumes that malicious model updates will be quite different compared to honest model, but this may not necessarily be the case for label-flipping attacks or backdoor attacks.
- RAGD assumes the existence of a static adjacency matrix, defining the edge weights between any two nodes. It also assumes that any attack edge has a weight of $0 < \epsilon < \frac{1}{2}$.
- We assume that nodes will not be fully connected.

C. Krum

Krum [47] attempts to improve the general Byzantine resilience in distributed machine learning. This approach operates on the premise that Byzantine model updates are prone to deviate from the updates produced by honest participants. More specifically, the aggregation involves computing a score $s(i)$ for every received model i, which corresponds to the sum of the squared distance between i and its $n - f - 2$ nearest distant-wise neighbours, where f corresponds to the maximum number of Byzantine participants. The model m with the lowest score, such that $m = \arg\min_i s(i)$, is chosen as the next model to train on.

IV. Threat model and preliminaries

A. Adversarial assumptions

Assumption 1. *The adversary is only capable of influencing the learning process through the predefined Decentralized Learning API.*

The adversary's only method of communicating with other nodes or influencing the learning process is through the default Decentralized Learning API to which honest nodes have access as well. The latter implies that the adversary does not have the ability to manipulate other nodes' local models or data.

Assumption 2. *Sybil model updates show high similarity compared to honest model updates.*

We assume that the model updates by Sybils exhibit a higher degree of similarity compared to updates made by honest participants, as stated by prior work [31].

Assumption 3. *The adversary is unrestricted in both the quantity of Sybil nodes it can create and the selection of honest nodes it can form attack edges to.*

Assumption 4. *The creation of Sybils by the adversary does not increase its adversarial computing capabilities.*

Regardless of the number of Sybil entities created by the adversary, we make the assumption that the computational capabilities of the adversary remain constant and do not scale with the number of Sybil entities. The primary rationale behind this assumption is the limited knowledge of the internal state of the model between the aggregation and training phases. More specifically, considering Figure 2, no participating node can with certainty determine the internal state of other participating nodes, including the aggregated model before training. As per Assumption 2, it follows that each Sybil must utilize the same aggregated model prior to training, thereby implying equivalence to Assumption 4.

B. Network restrictions

Assumption 5. $\exists\ e \in \mathbb{N}$ *such that* $d_i \leq e$, $\forall i \in N$, *where* d_i *represents the degree of node* i.

In order to restrict the impact that any individual node may exercise on the network, we assume the existence of an upper bound on the degree of any node. Such bounds may arise naturally in certain environments, such as peer-to-peer networks TODO SOURCE. This constraint serves to limit the potential harm that any one node may cause.

Enforcing an upper bound on the degree of nodes has been studied before. todo: find existing methods to achieve this.

Assumption 6. *Every node has at least one honest neighbour.*

C. Adversarial attack strategy

We define the most effective type of attack in similarity-based aggregation techniques in Decentralized Learning as the *Spread Sybil Poisoning Attacks* (SSP attack). That is, the adversary aims to evade detection by maximizing the distance between its attack edges while increasing the effect of the attack by minimizing the distance between any honest node and the nearest attack edge. The latter part of this problem resembles the *Maximal Covering Location Problem* [48], which is known to be an NP-Hard problem [49].

To determine attack edge positions for SSP attacks, we propose a simplified approach using the K-medoids unsupervised clustering algorithm, assigning attack edges to the medoids. Furthermore, we define a parameter for SSP attacks, ϕ, which represents the average density of attack edges per node. As such, if $\phi = 1$, we say that every honest node has exactly one attack edge. For other values of ϕ, each honest node receives $|N| \cdot \lfloor \phi \rfloor$ attack edges, where $|N|$ is the total number of honest nodes. The remainder, denoted as $\phi \bmod 1$, is distributed according to the K-medoids clustering algorithm. The resulting attack edge positions are then grouped and distributed over the Sybils while upholding Assumption 5.

V. Design of Sydle

Our solution, Sydle, attempts to deliver the same performance as FoolsGold in terms of accuracy and attack mitigation, while at the same time enjoying the scalability advantages offered by decentralized learning. This is achieved by reducing an adopted version of FoolsGold to a subfunction of our algorithm. Moreover, we integrate a probabilistic gossiping mechanism to for knowledge spreading. By doing so, the FoolsGold subfunction is capable of detecting distant attack edges from the same adversary.

In short, Sydle has been designed to handle different scenarios of the worst-case attack scenario, as discussed in Section IV-C. These scenarios are the following:

- $\phi >= 2$: Every honest node has at least two attack edges, whereas a Sybil node cannot form more than one attack edge to any given node. As a result, the honest node possesses the ability to gain knowledge of at least two distinct Sybil nodes. In such a situation, the adopted FoolsGold subfunction can detect Sybil nodes directly through its similarity mechanism, making it capable of mitigating the attack.
- $\epsilon < \phi < 2$: In the case where at least one node has fewer than 2 attack edges, it will no longer be capable of detecting Sybils among its direct neighbours. The Sybil updates it receives will likely vary from all other received updates and therefore considered honest as per Assumption 2. In this case, our gossiping mechanism acts as a medium for knowledge spreading. The update history of probabilistically selected nodes are propagated to neighbours, increasing nearby nodes' knowledge about nearby attack edges and TODO FINISH
- $\phi < \epsilon$: A low value of ϕ will result in sparse and distant attack edges. In this scenario, it is conceivable that the probabilistic gossiping mechanism may be unable to disseminate knowledge to an extent that enables all nodes with attack edges to detect the presence of Sybils. This phenomenon may manifest in two ways: firstly, the probabilistic gossip mechanism disseminates sufficient knowledge over a large period of time for attacked nodes

to have theoretical awareness of another distant attack edge. However, the received gossiped knowledge is likely to be significantly outdated, rendering the more recent knowledge from their direct attack edge to have diverged to such a degree that the Sybil is considered honest as per Assumption 2. Secondly, the probabilistic gossiping mechanism may be unable to provide an attacked node with the required knowledge to recognize an attack prior to the completion of the training process. In both aforementioned scenarios, we argue that due to a natural dampening effect originating from the train-aggregate loop on each node, depicted in Figure 2, an attack's effect will likely result in a negligible and tolerable effect on the average global model.

A. Adopting FoolsGold

Given that that Sybil updates are likely to closely resemble eachother, we gain the ability to compare the model updates received from our neighbours every round and decrease their influence during aggregation as their similarity to one or more neighbours grows. Note that nodes may often not have sufficient information to have a definite conclusion.

- Node's trust themselves
- include history of more nodes but only consider direct neighbours in aggregated model

This section assumes the reader is familiar with FoolsGold [31] (see Section III-A).

B. Probabilistic gossiping mechanism

The aforementioned adopted FoolsGold sub-function requires additional knowledge about its indirect neighbours to increase its detection rate, as the received model updates from direct neighbours is not sufficient to detect Sybils if the node is only connected to a single Sybil. To facilitate the knowledge spreading, we devised a probabilistic gossiping mechanism.

1) Probabilistic gossiping: First, let us define the method in which random model update histories are selected to be propagated to a neighbouring node. In short, Sydle uses a weighted random selection algorithm to select which node's model update history to propagate.

More specifically, let $\mathcal{H}_i$ denote the database of model update histories associated with node i. $\mathcal{H}_i$ consists of a list of tuples, with each tuple of the form $(p, h, r, d, f) \in \mathcal{H}_i$, where p represents the peer from which the model update history originates, h corresponds to the model update history, r is the round from which the model update history originates, d is the distance from node i to node p in the number of hops, and f is the neighbour of node i from which this model update history has been received. Note that node p's model update history h in round i is defined as the sum of all model updates originating from node p, i.e. $h_p^i = w_p^i + h_p^{i-1}$, given model update w_p^i from node p in round i. Given current node i and its neighboring node j, let the filtered database of model update histories $\mathcal{H}_i'$ be defined as $H_i' = \{(p, h, r, d, f) \mid (p, h, r, d, f) \in H_i, p \notin \{i, j\} \wedge f \neq j\}$. This filtered database is used for performing a weighted random selection of a model update history from node i to node j.

First, weights are assigned to the entries of the filtered database of model update histories. These weights directly correspond to the distance d and are assigned according to the exponential distribution:

$$P(d) = \lambda e^{-\lambda d} \tag{6}$$

where d is the distance in the number of hops between current node i and the node of which the weight is computed. λ can be considered a hyperparameter indicating the relevance of propagating the model update history of distant nodes. The selection of the exponential distribution is not arbitrary, as it prioritizes the propagation of the update history of nearby nodes over that of distant nodes. This approach mitigates distant attack edges by leveraging the natural dampening effect described previously. After the weights have been assigned to filtered dataset of model update histories, a weighted random selection is performed to select which model update history to propagate.

A node's local database of model update histories can be updated in the following ways:

- direct neighbour update
- haven't seen before
- is more up-to-date than the history we had previously

If scaling issues occur or training will take very long, we can consider dropping model update histories

Optimizations possible for future work.

Mention scaling attack somewhere? (Simply scale the model update 10 times)

Mention synchronous training rounds?

2) Secure and efficient communication: Sydle replaces the traditional model sharing discussed in Section II-B with a more sophisticated communication protocol. As the FoolsGold subfunction requires the model update history, we alter the communication mechanism to support this.

We assume that honest nodes and sybils both send all their neighbours the same updated model, but it would be possible to implement mechanisms to detect this.

gossip model: message consists of 2 parts, own signed history and other signed history.

1) Nodes train their model and then compute their own update history.
2) Nodes send their signed history to their neighbours.
3) Neighbours obtain a signed history of their neighbour and are capable of computing the model update by comparing the signed history with the signed history received in the prior round.
4) Neighbours can re-use the signed history by sending it to their neighbours according to the probabilistic gossiping mechanism.

As the FoolsGold subfunction requires the model history as well as the updated model in the aggregation, a traditional system would require

Table I: The datasets used in the evaluation of Sydle. TODO LAYOUT

Dataset	Model	Learning rate
CIFAR-10 [44]	LeNet-5	$\eta = 0.004$ [50]
SVHN [51]	LeNet-5	$\eta = 0.004$ [50]
FashionMNIST [46]	Single soft-max layer	$\eta = 0.01$ [31]
MNIST [52]	Single soft-max layer	$\eta = 0.01$ [31]

Note that we construct messages such that a message $m_{i \to j}$ from node i to j can be decomposed into $\langle h, g \rangle$, where h_i represents node i's signed updated model history to include the latest training round and g is the signed model history

todo: define the history of model updates as the sum of model updates.

3) Downtime support: Note that Sydle supports nodes going offline for an arbitrary period. When the offline node becomes available again, the model update can be computed by waiting an additional round while directly obtaining the model history.

C. Informativeness dilemma

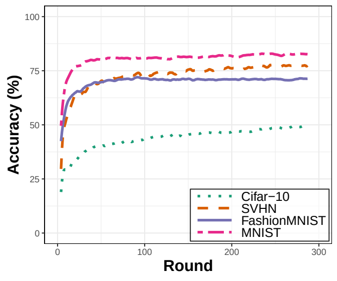
Explain that using Sydle introduces convergence delay, as it is possible to falsly detect positives among the set of honest neighbours due to high similarity with each other when they have a similar local dataset. However, our experiment show that eventually similar performance can be achieved.

VI. Evaluation

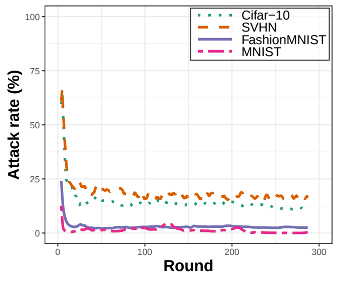
A. Experimental setup

Sydle was implemented in Python3 for experimental evaluation and is online available [?]. We have used the PyTorch [53] library for the training of machine learning models. As for the communication between the individual nodes, we leveraged IPv8 [?], which provides an API for constructing network overlays in order to simulate P2P networks. Furthermore, we adopted the Gumby [?] library as the experimental execution framework, which was specifically designed for sophisticated experiments with IPv8 involving many nodes. All experiments were performed on the Distributed ASCI Supercomputer 6 (DAS-6) [54]. Each node in the compute cluster has access to a dual 16-core CPU, 128 GB RAM and either an A4000 or A5000 GPU.

1) Datasets: The datasets that we used to evaluate can be found in Table I. These datasets were chosen for a number of reasons. First of all, MNIST [52] is a commonly used dataset in the evaluation of machine learning algorithms. FashionMNIST was developed as a more challenging variant of MNIST. The choice for SVHN and CIFAR-10 is motivated by the increased complexity of models required to obtain acceptable accuracy, which may affect the performance of Sydle. The usage of complex models in evaluation is often overlooked in other works TODO SOURCES [31], [39]. Note that all the datasets used in this experimental evaluation focus on image classification. We argue that that focusing on image classification is justifiable as it is known as a well-established task in machine learning. Furthermore, image classification frequently serves as a benchmark for evaluating novel distributed machine learning algorithms TODO SOURCES [31], [36], [39], and there exist a variety of widely available datasets specifically constructed for this task.

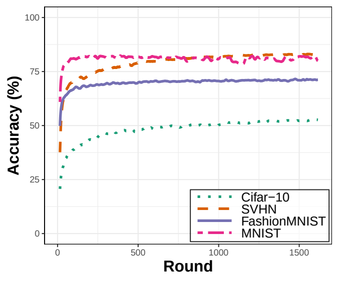


(a) Accuracy per dataset

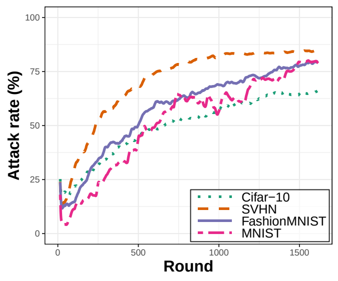


(b) Attack rate per dataset

Figure 7: Accuracy and attack rate for the label-flip attack on different datasets (300 rounds).



(a) Accuracy per dataset



(b) Attack rate per dataset

Figure 8: Accuracy and attack rate for the backdoor attack on different datasets (1600 rounds).

All models in Table I are trained using SGD.

2) Data distribution: Todo: discuss Non-IID data distributions. Argue that we use Dirichlet distribution

- Attacks:
  - Label-flipping attack
  - backdoor attack
- network topology is random geometric graph (should be in experimental setup, not here)
- network topology remains static during all experiments.
- table with default experiment parameters?
- the success of the attacker is measured by applying the dataset transform functions on the test data and compute the accuracy, we call this the attack rate. Note: we compute this metric directly after the aggregation step (see figure 2) and take the average of all nodes. Same for accuracy.

B. Effect of dataset

Notes for later:

- Figure 7 and 8 have been constructed with the following parameters;
  - Number of honest nodes: 99 for labelflip and 49 for backdoor.

- $\phi = 1$
- $\alpha = 0.1$

C. Comparison with other techniques

D. Effect of attack edge density

E. Effect of data distribution

VII. Analysis

TODO: use the very early composed quick analysis to compose a much larger and more comprehensive theoretical analysis.

Given that our novel gossip mechanism ensures that models travel at most s edges and the maximum degree of a node is d, then the amount of *useful* attack edges is equal to:

$$u = \frac{n}{\frac{1-d^{(s+1)}}{1-d}}$$

For example, if $s = 1$ and $d = 3$, then every attack edge has to be at least $2 * s + 1 = 3$ *hops* apart and every attack edge will spread the malicious model to $\frac{1-d^{(s+1)}}{1-d} = 4$ nodes.

However, this comes with a trade-off, as the amount of models any given node will to calculate the cosine similarity between grows with $\frac{1-d^{(s+1)}}{1-d}$ as well. With Sybil attacks, adversaries typically aim to exploit a certain network property through creating multiple entity. We say that Sybil attacks are of no use when the maximum of theoretically *useful* attack edges is smaller than the degree, such that $u < d$. Note that this implies that nodes will need to keep track of $\frac{n}{d}$ nodes. Note: when the amount of attack edges exceeds u, there must be at least one node in the network detecting the Sybil attack, manifesting sub-optimal results.

However, regarding the analysis performed above, such strict bounds on a Sybil attack are in practice not required, as all honest nodes keep training the model after aggregation, having a fading effect on the label-flipping attack or the backdoor attack as the distance increases from the sybil node.

VIII. Discussion

Talk about the lack of knowledge in Figure 2.

Adversaries with extensive computation power may well be able to train numerous models each round, violating assumption 4.

Existing methods of discovering convergence in DL and stopping as soon as it has been reached to limit the effect of the backdoor attack?

explain why we can't see the sybil updates (because we don't see the aggregation step)

More research required to observe the effect of adding random noise to irrelevant weights in the neural network.

IX. Conclusion

References

[1] E. V. Polyakov, M. S. Mazhanov, A. Y. Rolich, L. S. Voskov, M. V. Kachalova, and S. V. Polyakov, "Investigation and development of the intelligent voice assistant for the internet of things using machine learning," in *2018 Moscow Workshop on Electronic and Networking Technologies (MWENT)*, 2018, pp. 1–5.

[2] B. T.K., C. S. R. Annavarapu, and A. Bablani, "Machine learning algorithms for social media analysis: A survey," *Computer Science Review*, vol. 40, p. 100395, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1574013721000356

[3] X. Wang and Y. Wang, "Improving content-based and hybrid music recommendation using deep learning," in *Proceedings of the 22nd ACM International Conference on Multimedia*, ser. MM '14. New York, NY, USA: Association for Computing Machinery, 2014, p. 627–636. [Online]. Available: https://doi.org/10.1145/2647868.2654940

[4] S. A. Salloum, M. Alshurideh, A. Elnagar, and K. Shaalan, "Machine learning and deep learning techniques for cybersecurity: A review," in *Proceedings of the International Conference on Artificial Intelligence and Computer Vision (AICV2020)*, A.-E. Hassanien, A. T. Azar, T. Gaber, D. Oliva, and F. M. Tolba, Eds. Cham: Springer International Publishing, 2020, pp. 50–57.

[5] J. Prusa, T. M. Khoshgoftaar, and N. Seliya, "The effect of dataset size on training tweet sentiment classifiers," in *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, 2015, pp. 96–102.

[6] J. Hestness, S. Narang, N. Ardalani, G. F. Diamos, H. Jun, H. Kianinejad, M. M. A. Patwary, Y. Yang, and Y. Zhou, "Deep learning scaling is predictable, empirically," *CoRR*, vol. abs/1712.00409, 2017. [Online]. Available: http://arxiv.org/abs/1712.00409

[7] A. Goldsteen, G. Ezov, R. Shmelkin, M. Moffie, and A. Farkash, "Data minimization for gdpr compliance in machine learning models," *AI and Ethics*, pp. 1–15, 2021.

[8] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y. Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, A. Singh and J. Zhu, Eds., vol. 54. PMLR, 20–22 Apr 2017, pp. 1273–1282. [Online]. Available: https://proceedings.mlr.press/v54/mcmahan17a.html

[9] J. Janai, F. Güney, A. Behl, A. Geiger *et al.*, "Computer vision for autonomous vehicles: Problems, datasets and state of the art," *Foundations and Trends® in Computer Graphics and Vision*, vol. 12, no. 1–3, pp. 1–308, 2020.

[10] P. Navarro, C. Fernández, R. Borraz, and D. Alonso, "A machine learning approach to pedestrian detection for autonomous vehicles using high-definition 3d range data," *Sensors*, vol. 17, no. 12, p. 18, Dec 2016. [Online]. Available: http://dx.doi.org/10.3390/s17010018

[11] A. Hard, K. Rao, R. Mathews, F. Beaufays, S. Augenstein, H. Eichner, C. Kiddon, and D. Ramage, "Federated learning for mobile keyboard prediction," *CoRR*, vol. abs/1811.03604, 2018. [Online]. Available: http://arxiv.org/abs/1811.03604

[12] T. Yang, G. Andrew, H. Eichner, H. Sun, W. Li, N. Kong, D. Ramage, and F. Beaufays, "Applied federated learning: Improving google keyboard query suggestions," *CoRR*, vol. abs/1812.02903, 2018. [Online]. Available: http://arxiv.org/abs/1812.02903

[13] M. Chen, R. Mathews, T. Ouyang, and F. Beaufays, "Federated learning of out-of-vocabulary words," *CoRR*, vol. abs/1903.10635, 2019. [Online]. Available: http://arxiv.org/abs/1903.10635

[14] Y. Cheng, Y. Liu, T. Chen, and Q. Yang, "Federated learning for privacy-preserving ai," *Communications of the ACM*, vol. 63, no. 12, pp. 33–36, 2020.

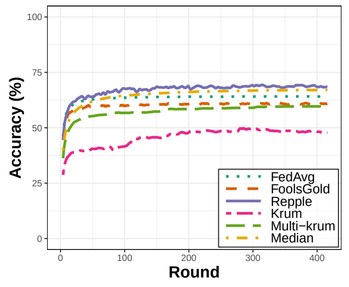
[15] L. Lyu and C. Chen, "A novel attribute reconstruction attack in federated learning," *CoRR*, vol. abs/2108.06910, 2021. [Online]. Available: https://arxiv.org/abs/2108.06910

[16] H. Yang, M. Ge, K. Xiang, and J. Li, "Using highly compressed gradients in federated learning for data reconstruction attacks," *IEEE Transactions on Information Forensics and Security*, vol. 18, pp. 818–830, 2023.

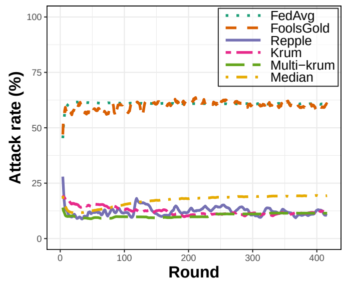
[17] H. S. Sikandar, H. Waheed, S. Tahir, S. U. R. Malik, and W. Rafique, "A detailed survey on federated learning attacks and defenses," *Electronics*, vol. 12, no. 2, 2023. [Online]. Available: https://www.mdpi.com/2079-9292/12/2/260

[18] P. Qiu, X. Zhang, S. Ji, Y. Pu, and T. Wang, "All you need is hashing: Defending against data reconstruction attack in vertical federated learning," 2022. [Online]. Available: https://arxiv.org/abs/2212.00325

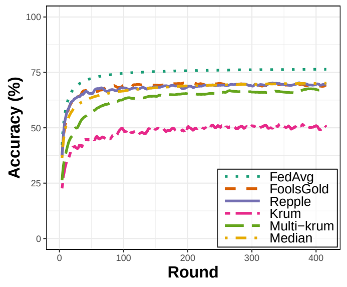
[19] J. Hamer, M. Mohri, and A. T. Suresh, "FedBoost: A communication-efficient algorithm for federated learning," in *Proceedings of the 37th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, H. D. III and A. Singh, Eds., vol.



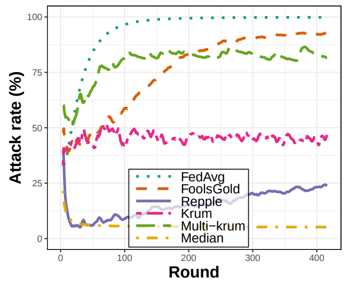
(a) Accuracy label-flip



(b) Attack rate label-flip

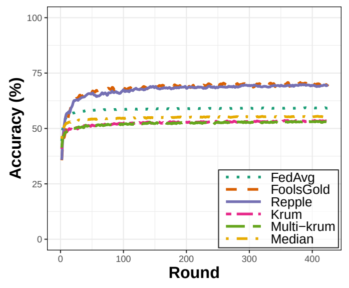


(c) Accuracy backdoor

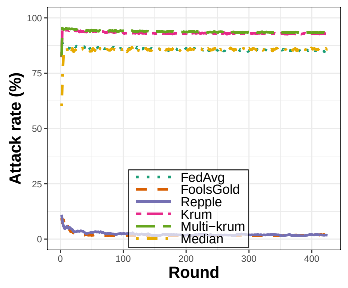


(d) Attack rate backdoor

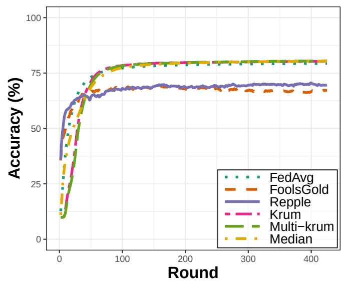
Figure 9: Comparison of Sydle against different techniques on $\phi = 1$.



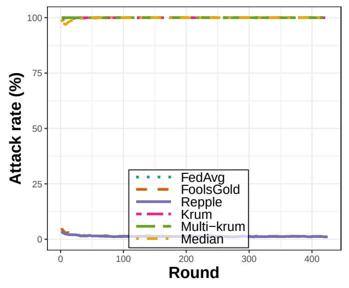
(a) Accuracy label-flip



(b) Attack rate label-flip



(c) Accuracy backdoor



(d) Attack rate backdoor

Figure 10: Comparison of Sydle against different techniques on $\phi = 4$.

119. PMLR, 13–18 Jul 2020, pp. 3973–3983. [Online]. Available: https://proceedings.mlr.press/v119/hamer20a.html

[20] S. Kadhe, N. Rajaraman, O. O. Koyluoglu, and K. Ramchandran, "Fastsecagg: Scalable secure aggregation for privacy-preserving federated learning," *CoRR*, vol. abs/2009.11248, 2020. [Online]. Available: https://arxiv.org/abs/2009.11248

[21] Y. Qi, M. S. Hossain, J. Nie, and X. Li, "Privacy-preserving blockchain-based federated learning for traffic flow prediction," *Future Generation Computer Systems*, vol. 117, pp. 328–337, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X2033065X

[22] I. Hegedűs, G. Danner, and M. Jelasity, "Decentralized learning works: An empirical comparison of gossip learning and federated learning," *Journal of Parallel and Distributed Computing*, vol. 148, pp. 109–124, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0743731520303890

[23] J. Hou, F. Wang, C. Wei, H. Huang, Y. Hu, and N. Gui, "Credibility assessment based byzantine-resilient decentralized learning," *IEEE Transactions on Dependable and Secure Computing*, pp. 1–12, 2022.

[24] V. Tolpegin, S. Truex, M. E. Gursoy, and L. Liu, "Data poisoning attacks against federated learning systems," in *Computer Security – ESORICS 2020*, L. Chen, N. Li, K. Liang, and S. Schneider, Eds. Cham: Springer International Publishing, 2020, pp. 480–501.

[25] N. M. Jebreel, J. Domingo-Ferrer, D. Sánchez, and A. Blanco-Justicia, "Defending against the label-flipping attack in federated learning," 2022. [Online]. Available: https://arxiv.org/abs/2207.01982

[26] D. Li, W. E. Wong, W. Wang, Y. Yao, and M. Chau, "Detection and mitigation of label-flipping attacks in federated learning systems with kpca and k-means," in *2021 8th International Conference on Dependable Systems and Their Applications (DSA)*, 2021, pp. 551–559.

[27] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," in *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, S. Chiappa and R. Calandra, Eds., vol. 108. PMLR, 26–28 Aug 2020, pp. 2938–2948. [Online]. Available: https://proceedings.mlr.press/v108/bagdasaryan20a.html

[28] Z. Sun, P. Kairouz, A. T. Suresh, and H. B. McMahan, "Can you really backdoor federated learning?" *CoRR*, vol. abs/1911.07963, 2019. [Online]. Available: http://arxiv.org/abs/1911.07963

[29] C. Wu, X. Yang, S. Zhu, and P. Mitra, "Mitigating backdoor attacks in federated learning," *CoRR*, vol. abs/2011.01767, 2020. [Online]. Available: https://arxiv.org/abs/2011.01767

[30] J. R. Douceur, "The sybil attack," in *Peer-to-Peer Systems*, P. Druschel, F. Kaashoek, and A. Rowstron, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 251–260.

[31] C. Fung, C. J. M. Yoon, and I. Beschastnikh, "Mitigating sybils in federated learning poisoning," *CoRR*, vol. abs/1808.04866, 2018. [Online]. Available: http://arxiv.org/abs/1808.04866

[32] Y. Mao, D. Data, S. Diggavi, and P. Tabuada, "Decentralized learning robust to data poisoning attacks," in *2022 IEEE 61st Conference on Decision and Control (CDC)*, 2022, pp. 6788–6793.

[33] C. Hu, J. Jiang, and Z. Wang, "Decentralized federated learning: A segmented gossip approach," *CoRR*, vol. abs/1908.07782, 2019. [Online]. Available: http://arxiv.org/abs/1908.07782

[34] I. Hegedűs, G. Danner, and M. Jelasity, "Decentralized learning works: An empirical comparison of gossip learning and federated learning," *Journal of Parallel and Distributed Computing*, vol. 148, pp. 109–124, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0743731520303890

[35] Z. Tang, S. Shi, B. Li, and X. Chu, "Gossipfl: A decentralized federated learning framework with sparsified and adaptive communication," *IEEE Transactions on Parallel and Distributed Systems*, vol. 34, no. 3, pp. 909–922, 2023.

[36] M. de Vos, A. Dhasade, A.-M. Kermarrec, E. Lavoie, and J. Pouwelse, "Modest: Bridging the gap between federated and decentralized learning with decentralized sampling," 2023.

[37] I. Hegedűs, G. Danner, and M. Jelasity, "Gossip learning as a decentralized alternative to federated learning," in *Distributed Applications and Interoperable Systems*, J. Pereira and L. Ricci, Eds. Cham: Springer International Publishing, 2019, pp. 74–90.

[38] A. G. Roy, S. Siddiqui, S. Pölsterl, N. Navab, and C. Wachinger, "Braintorrent: A peer-to-peer environment for decentralized federated learning," *CoRR*, vol. abs/1905.06731, 2019. [Online]. Available: http://arxiv.org/abs/1905.06731

[39] J. Verbraeken, M. de Vos, and J. Pouwelse, "Bristle: Decentralized federated learning in byzantine, non-i.i.d. environ-

ments," *CoRR*, vol. abs/2110.11006, 2021. [Online]. Available: https://arxiv.org/abs/2110.11006

[40] S. Guo, T. Zhang, H. Yu, X. Xie, L. Ma, T. Xiang, and Y. Liu, "Byzantine-resilient decentralized stochastic gradient descent," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 32, no. 6, pp. 4096–4106, 2022.

[41] J. Hou, F. Wang, C. Wei, H. Huang, Y. Hu, and N. Gui, "Credibility assessment based byzantine-resilient decentralized learning," *IEEE Transactions on Dependable and Secure Computing*, pp. 1–12, 2022.

[42] B. N. Levine, C. Shields, and N. B. Margolin, "A survey of solutions to the sybil attack," *University of Massachusetts Amherst, Amherst, MA*, vol. 7, p. 224, 2006.

[43] D. N. Tran, B. Min, J. Li, and L. Subramanian, "Sybil-resilient online content voting." in *NSDI*, vol. 9, no. 1, 2009, pp. 15–28.

[44] A. Krizhevsky, V. Nair, and G. Hinton, "Cifar-10 (canadian institute for advanced research)." [Online]. Available: http://www.cs.toronto.edu/ kriz/cifar.html

[45] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[46] H. Xiao, K. Rasul, and R. Vollgraf. (2017) Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. Https://github.com/zalandoresearch/fashion-mnist. [Online]. Available: https://github.com/zalandoresearch/fashion-mnist

[47] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, and J. Stainer, "Machine learning with adversaries: Byzantine tolerant gradient descent," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017.

[48] R. Church and C. ReVelle, "The maximal covering location problem," in *Papers of the regional science association*, vol. 32, no. 1. Springer-Verlag Berlin/Heidelberg, 1974, pp. 101–118.

[49] N. Megiddo, E. Zemel, and S. L. Hakimi, "The maximum coverage location problem," *SIAM Journal on Algebraic Discrete Methods*, vol. 4, no. 2, pp. 253–261, 1983. [Online]. Available: https://doi.org/10.1137/0604028

[50] C. Thapa, M. A. P. Chamikara, and S. Camtepe, "Splitfed: When federated learning meets split learning," *CoRR*, vol. abs/2004.12088, 2020. [Online]. Available: https://arxiv.org/abs/2004.12088

[51] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," in *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011. [Online]. Available: http://ufldl.stanford.edu/housenumbers/nips2011_housenumbers.pdf

[52] L. Deng, "The mnist database of handwritten digit images for machine learning research," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.

[53] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[54] H. Bal, D. Epema, C. de Laat, R. van Nieuwpoort, J. Romein, F. Seinstra, C. Snoek, and H. Wijshoff, "A medium-scale distributed system for computer science research: Infrastructure for the long term," *Computer*, vol. 49, no. 05, pp. 54–63, may 2016.