# Towards Sybil Resilience in Decentralized Learning

**Thomas Werthenbach**

| | |
|---|---|
| Student number: | 4772466 |
| Thesis committee: | Dr. ir. J.A. Pouwelse (supervisor) |
| | Dr. D.M.J. Tax |

4 July 2023

To obtain the degree of Master of Science in Computer Science
Software Technology Track
To be defended publicly on July 4, 2023

**Delft University of Technology**
Faculty of Electrical Engineering, Mathematics & Computer Science
Distributed Systems Group

# Towards Sybil Resilience in Decentralized Learning

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

**— MSc. Thesis —**

***Abstract*—Federated learning is a privacy-enforcing machine learning technology but suffers from limited scalability. This limitation mostly originates from the internet connection and memory capacity of the central parameter server, and the complexity of the model aggregation function. Decentralized learning has recently been emerging as a promising alternative to federated learning. This novel technology eliminates the need for a central parameter server by decentralizing the model aggregation across all participating nodes. Numerous studies have been conducted on improving the resilience of federated learning against poisoning and Sybil attacks, whereas the resilience of decentralized learning remains largely unstudied. This research gap serves as the main motivator for this study, in which our objective is to improve the Sybil poisoning resilience of decentralized learning.**

**We present SybilWall, an innovative algorithm focused on increasing the resilience of decentralized learning against targeted Sybil poisoning attacks. By combining a Sybil-resistant aggregation function based on similarity between Sybils with a novel probabilistic gossiping mechanism, we establish a new benchmark for scalable, Sybil-resilient decentralized learning.**

**A comprehensive empirical evaluation demonstrated that SybilWall outperforms existing state-of-the-art solutions designed for federated learning scenarios and is the only algorithm to obtain consistent accuracy over a range of adversarial attack scenarios. We also found SybilWall to diminish the utility of creating many Sybils, as our evaluations demonstrate a higher success rate among adversaries employing fewer Sybils. Finally, we suggest a number of possible improvements to SybilWall and highlight promising future research directions.**

***Index Terms*—Decentralized applications, Adversarial machine learning, Federated learning, Decentralized learning, Sybil attack, Poisoning attack**

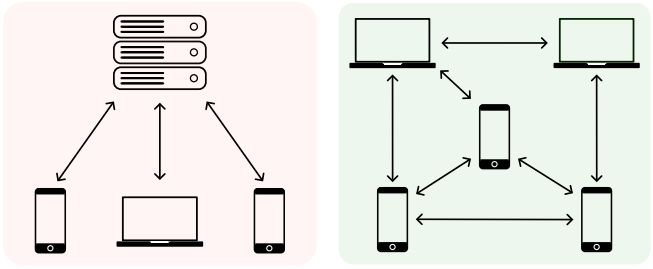
## I. Introduction

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], cybersecurity [2], and recommendations on social media [3] and music [4]. However, accurate machine learning models require large training datasets [5], [6], which can be difficult to collect due to privacy concerns [7] and recent privacy legislation [8]. *Federated learning* [9] has become a promising option for distributed machine learning. It has been proposed for the training of numerous industrial machine learning models [10]–[14]. Moreover, federated learning ensures the protection of privacy, as the user's data will not leave their device.

In contrast to centralized machine learning, model training in federated learning takes place on end-users' personal devices, which are often referred to as *edge devices* or *nodes*. The resulting trained models are communicated to a central server, commonly referred to as the *parameter server*, which aggregates these trained models into a single global model. 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 [15]. Although there exist attacks in which training data can be reconstructed based on model gradients [16], [17], defense mechanisms against this attack have been proposed [18], [19].

However, federated learning suffers from some disadvantages. First, the parameter server downloads the models of all participating nodes and broadcasts the aggregated global model each training round, inducing high communication costs and a potential bottleneck in the learning process. This may affect the overall convergence time [20]. Second, the scalability of the chosen aggregation function in terms of the number of nodes may vary greatly. In robust and secure federated learning aggregation methods, the incorporation of additional nodes during aggregation can result in a significantly increased computational effort for the parameter server [21]. Third, the parameter server performing the aggregation poses a single point of failure [22], [23]. Disruptions to the parameter server can cause downtime and hinder the overall model training process, particularly when nodes require the globally aggregated model before proceeding the training. An upcoming alternative that aims to resolve these issues is *decentralized learning* [24]–[27], 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. Each node in this network individually performs the aggregation using their neighbors' models (Figure 1). This alleviates the scalability limitations and single point of failure issues imposed on federated learning and paves the path for boundless scalability. While the information available during aggregation is more limited relative to federated learning, decentralized learning has the potential to obtain similar results compared to federated learning [28].

Although decentralized learning resolves the drawbacks faced by federated learning, it is still vulnerable to malicious 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 com-



(a) Federated learning (b) Decentralized learning

Figure 1: Exemplary network topologies in federated learning and decentralized learning.

pared to federated learning. This causes decentralized learning to have relatively lower resistance against possible poisoning attacks [29]. Poisoning attacks can generally be classified in two categories, namely those of *targeted poisoning attacks* and *untargeted poisoning attacks*. Targeted poisoning attacks focus on a specific goal that an adversary aims to achieve, while untargeted poisoning attacks aim to hinder the result of the training process without any particular goal in mind. The effect of these attacks can often be amplified by combining them with a Sybil attack [30], in which an adversary creates a substantial number of virtual nodes to increase its influence. As such, an adversary may deploy the Sybil attack to spread their poisoned model more rapidly 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 Sybil poisoning attacks in distributed machine learning has mainly been done in federated learning settings. One popular example of such work is *FoolsGold* [31], which aims to increase resilience against targeted Sybil poisoning attacks under the assumption that all Sybils will exhibit highly similar behavior. Experimental results suggest that FoolsGold can provide effective protection against Sybil attacks in small-scale federated learning.

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 SybilWall, which shows significant resilience towards defending against targeted poisoning attacks while enjoying the boundless scalability offered by decentralized learning. More specifically, we achieve this by introducing a probabilistic gossiping mechanism for data dissemination. We performed an empirical evaluation of SybilWall and found that it achieved satisfactory accuracy, convergence rate, and Sybil poisoning resilience on 4 different datasets. Moreover, comparative evaluations demonstrate SybilWall's superior Sybil resilience over numerous existing solutions. Lastly, we found that SybilWall successfully diminishes the utility of creating many Sybils.

To the best of our knowledge, this work is the first to propose a defensive algorithm against poisoning attacks amplified by the Sybil attack in decentralized learning. In short, our contributions are the following:

- We define the Spread Sybil Poisoning attack in Section IV for effective Sybil poisoning attacks in decentralized learning and decompose it into three distinct scenarios.
- We present SybilWall, a pioneering algorithm for Sybil poisoning resilience with boundless scalability in decentralized learning, in Section V.
- We performed an empirical evaluation of the performance of SybilWall in Section VI on various datasets and against competitive alternatives.

## II. Background

Federated learning was initially proposed by Google [9] as a means of training machine learning models on real user data without compromising user privacy. However, federated learning is associated with limitations in scalability. Decentralized learning is a promising alternative as it resolves scalability limitations through decentralization. Both distributed machine learning technologies are prone to poisoning attacks, of which the effects can be amplified by employing the Sybil attack.

### A. Federated learning

Federated learning achieves privacy-enforcing machine learning by training all machine learning models on the edge devices (nodes) of the participating users, containing real user data (Figure 1a). Training proceeds in synchronous rounds. During each training round, the participating nodes train the globally shared model on their private data for a predefined number of epochs and send the trained models to a central parameter server. The role of the parameter server is to aggregate all trained models into a global model without the need for the training data. After aggregation, the parameter server communicates the global model to all nodes, immediately followed by the start of the next training round. Alternatively, nodes may send *gradients* to the parameter server rather than the trained model. These gradients are the result of training the model on the node's local dataset, e.g. with stochastic gradient descent (SGD), in that particular training round. These gradients can be used to compute the trained model by adding the gradients to the aggregated global model.

The original federated learning paper [9] 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 $\mathcal{D} = \bigcup_{j \in N} \mathcal{D}_j$.

The goal 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.

In federated learning, all participating nodes are only connected to the parameter server, such that the network graph $\mathcal{G}$ is defined as a tuple of nodes and undirected edges $\langle N, E \rangle$, for which there exists a one-to-one mapping $N \rightarrow E$, such that for parameter server $s$, $\forall n \in N, \langle n, s \rangle \in E$.

### B. Decentralized learning

Decentralized learning is an upcoming alternative to federated learning [24]–[27] (Figure 1b). In contrast to federated learning, which relies on a parameter server to aggregate locally trained models, aggregation in decentralized learning takes place on a smaller scale, as it is performed by every participating node. During each training round, every node trains their local aggregated model on their private data for a predefined number of epochs and then broadcasts only the trained model to its neighboring nodes. After receiving the trained models from all neighbors, every node aggregates both its own and its neighbors' trained models to construct the aggregated intermediary model, which is then used to train on during the next training round. This train-aggregate loop is depicted in Figure 2.

By eliminating the need for a central parameter server, decentralized learning does not suffer from the scalability limitations encountered in federated learning. These improvements in scalability can be decomposed into three distinct aspects:

1) *Communication costs*: In federated learning, all models are downloaded and uploaded by the parameter server every training round, forming a communication bottleneck bounded by the parameter server's internet connection. Such bottlenecks are reduced in decentralized learning depending on a node's number of neighbors.
2) *Memory*: Storing all models in memory during aggregation may result in substantial memory usage. In decentralized learning, aggregation coincides with a significantly reduced number of models compared to federated learning, thus diminishing memory-related limitations.
3) *Aggregation time*: The time complexity of the more sophisticated aggregation functions may not scale linearly with respect to the number of participating nodes. Due to the decentralization of the aggregation, the number of models in each aggregation is greatly reduced.

Furthermore, nodes likely do not have access to all information in the network. This may cause an impaired convergence rate compared to federated learning [27] and lead to increased vulnerability to potential attacks [23]. However, decentralized learning has been shown to obtain comparable performance results compared to federated learning [28] and may sporadically outperform federated learning altogether [32], [33].

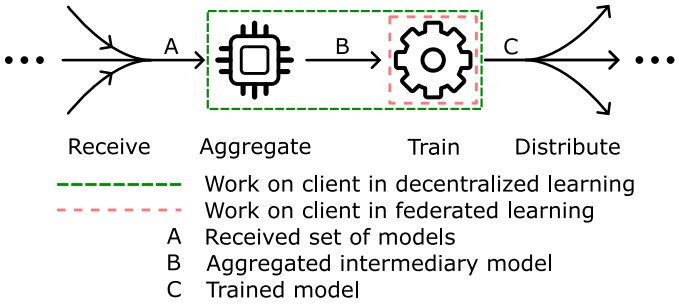



Figure 2: The general train-aggregate loop executed by all nodes participating in decentralized learning. The difference between the work performed by nodes in federated learning and decentralized learning is highlighted.

A network graph $\mathcal{G}$ in decentralized learning can be defined as the tuple $\langle N, E \rangle$, where $N$ and $E$ represent the set of nodes and the set of undirected edges, respectively. The set of edges $E$ is restricted such that $\forall \langle n, m \rangle \in E$ it holds that $n \in N \wedge m \in N \wedge n \neq m$. We say that nodes $n$ and $m$ are neighbors if $\langle n, m \rangle \in E$. If $\langle n, m \rangle \notin E$, $n$ is a distant node (or indirect neighbor) in $m$'s perspective and vice versa. Moreover, the distance between any two nodes is defined as the minimum number of edges connecting the two nodes. In contrast to federated learning, the network graph $\mathcal{G}$ in decentralized learning does not contain a parameter server.

### C. Targeted poisoning attacks

A *poisoning attack* is a type of Byzantine attack, which encapsulates all methods by which an adversary may attempt to compromise the integrity of the global model in decentralized learning and federated learning. Poisoning attacks typically include *data poisoning* or *model poisoning*. *Data poisoning* involves malicious alteration of the training data, while *model poisoning* entails adapting the process in which the model is trained [29]. In this work, we only consider data poisoning. Furthermore, poisoning attacks can be subdivided into two categories: *targeted poisoning attacks* and *untargeted poisoning attacks*. With untargeted poisoning attacks, the adversary aims to decrease the performance metric of the global model without any particular goal in mind. On the other hand, targeted poisoning attacks are employed to achieve a specific goal by manipulating the global model to behave in a deterministic manner that deviates from objectively correct behavior. In this paper, we exclusively consider two types of targeted poisoning attacks: the label-flipping attack [34], [35] and the backdoor attack [36]–[38].

The label-flipping attack can be deployed as an attempt to increase the probability of two targeted classes being 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 an arbitrary sample belonging to class $t_1$ is more likely to be classified as class $t_2$ by the global model and vice versa. One logical way of achieving this through data poisoning is by explicitly transforming the adversary's

local dataset $\mathcal{D}$ into 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:

$$\begin{aligned}\mathcal{D}' = &\{(x,y) \in \mathcal{D} \mid y \neq t_1 \wedge y \neq t_2\} \\ &\cup \{(x,t_1) \mid (x,y) \in \mathcal{D},\ y = t_2\} \\ &\cup \{(x,t_2) \mid (x,y) \in \mathcal{D},\ y = t_1\}\end{aligned} \tag{3}$$

The backdoor attack requires a more sophisticated 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 a small square or triangle in the top left corner of the input image [39]. Given a target class $t$ and a function $f$ that introduces a hidden pattern to input samples, the transformation applied on the adversary's local dataset $\mathcal{D}$ can be defined as:

$$\mathcal{D}' = \{(f(x), t) \mid (x,y) \in \mathcal{D}\} \tag{4}$$

### D. The Sybil attack

The Sybil attack [30] is an adversarial strategy in distributed environments in which the attacker exploits the inability to verify the authenticity of any node's identity. Through the effortless creation of fake nodes, *Sybils*, and strategical edges to honest nodes, the attacker may gain significantly more influence compared to honest nodes. We denote the edges between Sybils and honest nodes as *attack edges*. A typical scenario in which the Sybil attack may be deployed is *majority voting* [40], [41]. In such a case, an attacker can trivially generate sufficient Sybils to outnumber all honest voters.

Methods for mitigating the Sybil attack through an admission control system to the decentralized network have been proposed [42]–[44], but are often not frictionless or are based on an invite-only system. Adoption of such systems may take place at a slower rate due to its decreased accessibility and usability [45]. The importance of frictionless admission becomes increasingly apparent considering that decentralized learning can be implemented as a background task [13].

A network graph on which a Sybil attack is deployed can be defined as $\mathcal{G} = \langle N', E' \rangle$, such that $N' = N \cup \mathcal{S}$, where $\mathcal{S}$ is the unbounded set of Sybils created by the adversary. Note that Sybils and honest nodes are indistinguishable from the typical point of view. The modified set of edges $E'$ is defined as $E' = E \cup E_{\mathcal{S}}$, where $E_{\mathcal{S}}$ is the set of attack edges and edges between Sybils, which is highly dependent on the strategy of the adversary. Note that attack edges always consist of at least one Sybil, such that $\forall \langle i,j \rangle \in E_{\mathcal{S}}, i \in \mathcal{S} \vee j \in \mathcal{S}$.

In this work, we consider the targeted Sybil poisoning attack, in which an adversary aims to amplify the effects of a targeted poisoning attack by creating Sybils. These Sybils help spread the adversary's malicious model more rapidly and effectively throughout the network.

## III. Related work

Numerous studies have been conducted in order to improve poisoning resilience in a form of distributed machine learning. This section provides an overview of two existing defense mechanisms used in the empirical evaluation of SybilWall.

### A. FoolsGold

FoolsGold [31] is an algorithm designed to mitigate targeted Sybil poisoning attacks in federated learning settings. It builds on the assumption that Sybil model gradients show a substantially higher degree of similarity relative to that of honest model gradients, as they collaborate to reach the goal of a targeted poisoning attack. By computing the similarity between these model gradients, FoolsGold manages to successfully mitigate Sybil poisoning attacks in federated learning.

During aggregation, the parameter server first computes the pairwise cosine similarity score for all gradient histories. The gradient history of node $i$ in round $T$ is defined as $h_i^T = \sum_{t=0}^{T} g_i^t$, where $g_i^t$ are the gradients of a model obtained by training the model on node $i$ in round $t$. However, as honest nodes may still produce similar gradient histories, this may result in an increased number of false positives. In an effort to decrease the number of false positives, FoolsGold implements a *pardoning* mechanism. This pardoning mechanism multiplies each similarity score $s_{ij}$ by the ratio of the maximum score of node $i$ and the maximum score of node $j$ in the cases where the latter is greater, such that $s_{ij}$ is multiplied by $\frac{\max_v s_{iv}}{\max_v s_{jv}}$ if $\max_v s_{iv} < \max_v s_{jv}$.

Subsequently, the scores are aggregated for each node by taking the complement of the maximum score, such that node $i$'s aggregated score $s_i'$ can be defined as $s_i' = 1 - \max_v s_{iv}$. These aggregated scores now represent the extent to which a node can be trusted, based on its cosine similarity score. The aggregated scores are then rescaled such that the highest aggregated score equals 1, as FoolsGold assumes the existence of at least one honest node. Each score now indicates a node's similarity to any other node, with a value close to 0 indicating high similarity, while a value near 1 suggests little similarity. These scores are then transformed using a bounded logit function to prioritize higher-scoring nodes. Finally, the scores are normalized and adopted as weights in a weighted average on the trained models to compute the aggregated model.

A reproduction of FoolsGold's results can be found in Figure 3, where the attack score 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 FedAvg. However, as discussed in Section II-B, federated learning can be considered unscalable as the number of participating nodes increases. The limited scalability of FoolsGold is further highlighted by its $\mathcal{O}(n^2)$ pairwise cosine similarity computation and the memory capacity required to store these models. Figure 4 demonstrates the $\mathcal{O}(n^2)$ time complexity of the pairwise cosine similarity computation on the LeNet-5 model [46]. We further note that the experiments required to generate Figure 4 consumed the maximum memory

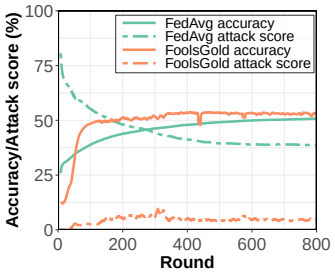



Figure 3: FoolsGold and FedAvg in federated learning setting using the CIFAR-10 [47] dataset on a LeNet-5 [46] model.

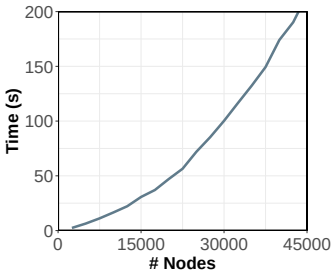



Figure 4: Pairwise cosine similarity computation time against the number of nodes (LeNet-5 [46]).

allocated. This emphasizes the memory limitation associated with federated learning. Furthermore, Figure 5 shows the performance of FoolsGold in a decentralized setting against the performance of our improved solution, SybilWall, based on FoolsGold's intuitions. When comparing both Figures 5a and 5b, it becomes clear that FoolsGold's performance heavily depends on the network topology, while SybilWall demonstrates relatively higher and more consistent Sybil resilience.

### B. Krum

Krum [48] attempts to improve the overall Byzantine resilience in distributed machine learning. This approach operates on the assumption that Byzantine model gradients are prone to deviate from the gradients produced by honest nodes. More specifically, the aggregation involves computing a score $s(w)$ for every received model $w$. This score corresponds to the sum of the squared distances between $i$ and its $n - f - 2$ nearest neighbours, where $f$ corresponds to the maximum number of Byzantine nodes Krum is configured to protect against. Finally, the model $m$ with the lowest score, such that $m = \arg\min_w s(w)$, is chosen as the next global model.

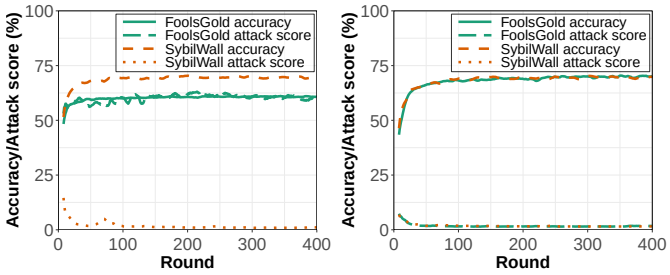
## IV. Threat Model and Assumptions

This section provides an overview of the assumptions and threat model used throughout this work.

### A. Adversarial assumptions

**Assumption 1.** *The adversary can only communicate with other nodes through the default decentralized learning API.*

As the adversary can only communicate with other nodes through the default decentralized learning API, it does not




(a) Network topology 1 (b) Network topology 2

Figure 5: FoolsGold and SybilWall in decentralized learning using the FashionMNIST [49] dataset on a single-layer softmax neural network.

possess the ability to directly manipulate other nodes' local models or data. However, adversaries are not restricted in manipulating their own model, which is sent to neighbors. We also assume that the decentralized learning API enforces homogeneous model broadcasting. That is, every node broadcasts the same model to each of its neighbors every training round. In practice, this can be enforced by adopting existing algorithms [50]. Lastly, we assume that the default decentralized learning API adopts the use of signatures to prevent spoofing.

**Assumption 2.** *All used cryptographic primitives are secure.*

The signatures used by the decentralized learning API, as well as any other cryptographic primitives employed throughout this work, are assumed to be secure.

**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.** *Sybil models show high similarity compared to honest models.*

Given the context of targeted poisoning attacks, Sybils are created by an adversary to achieve a specific goal during decentralized learning. As these Sybils share their training dataset, their trained models will likely show a high similarity.

In contrast to prior work [31], we assume a high similarity between the trained models of Sybils, rather than the model gradients, i.e. the difference between the aggregated intermediary model and the trained model. Due to the lack of knowledge of the aggregated intermediary model between the aggregation and training stages (Figure 2), no node can ascertain the model gradients of another node in decentralized learning.

**Assumption 5.** *The creation of Sybils by the adversary does not increase its adversarial computing capabilities.*

Following Assumption 4 and the lack of knowledge of the aggregated intermediary model, we must assume that each Sybil utilizes the same aggregated intermediary model. This assumption is enforced through Assumption 5, that is, the adversary does not have sufficient adversarial computing capabilities to execute the train-aggregate loop for each Sybil each round.

### B. Network restrictions

**Assumption 6.** *$\exists\ e \in \mathbb{N}$ such that $d_i \leq e$, $\forall i \in N$, where $d_i$ represents the degree of node $i$.*

We restrict the impact that any individual node may exercise on the network, by assuming existence of an upper bound on the degree of any node. Such bounds may arise naturally due to internet connection speeds, but may also be detected through existing algorithms. For example, a network latency-based avoidance mechanism [51] can be used to discover multiple edges of a node. Another alternative is to perform a random walk or a breadth-first search, which are known to be biased toward high-degree nodes [52].

**Assumption 7.** *Every node has at least one honest neighbour.*

This final assumption is inherited from prior work [31], as the cosine similarity function requires a baseline for *honest* work for measuring relative similarity. This might be achieved through an invite-only network with accountability [43]. We note that Eclipse attacks [53] are out of the scope of this work.

### C. Adversarial strategy

We define an intuitive and effective type of worst-case attack in similarity-based aggregation techniques in decentralized learning as *Spread Sybil Poisoning Attacks* (SSP attacks). That is, the adversary aims to avoid detection by maximizing the distance between its attack edges. At the same time, the adversary attempts to increase the influence 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* [54], which is known to be an NP-Hard problem [55]. To determine the attack edge positions for SSP attacks, we propose a heuristic approach using the unsupervised clustering algorithm K-medoids [56], assigning attack edges to the medoids.

Furthermore, we define a parameter for SSP attacks, $\phi$, which represents the average density of attack edges per node. Note that the attack edges are as spread out as possible, such that $\forall a_i, a_j \in \mathcal{A}, |a_i - a_j| \leq 1$, where $\mathcal{A}$ represents the set of the number of attack edges per node. For any value of $\phi$, each honest node receives $\lfloor\phi\rfloor$ or $\lceil\phi\rceil$ attack edges. Therefore, the total number of attack edges is denoted as $\lceil|N| \cdot \phi\rceil$. The remainder, defined by $\phi \bmod 1$, is distributed according to the K-medoids clustering algorithm. The resulting attack edge positions are then grouped and distributed over the Sybils while maintaining Assumption 6. We define three attack scenarios for specific ranges of $\phi$. These attack scenarios are the following:

i. *Dense Sybil poisoning attack*. $\phi \geq 2$. Every honest node has at least two attack edges, whereas any distinct Sybil cannot form more than one attack edge to any given node. As a result, each honest node is a direct neighbor of at least two distinct Sybils.
ii. *Distributed Sybil poisoning attack*. $\epsilon < \phi < 2$. There exists at least one node which is connected to fewer than 2 attack edges and will therefore only be connected to at most one Sybil.
iii. *Sparse Sybil poisoning attack*. $\phi \leq \epsilon$. A low $\phi$ will result in sparse and distant attack edges. Any node has a probability of $\phi$ of being directly connected to a Sybil.

## V. Design of SybilWall

Our proposed algorithm, SybilWall, takes inspiration from federated learning but was meticulously designed to enable boundless scalability through decentralization. Another primary purpose of SybilWall is the mitigation of targeted Sybil poisoning attacks. We used the state-of-the-art FoolsGold algorithm (federated learning) as a starting point for SybilWall.

A simplified overview of the architecture of SybilWall can be found in Figure 6. During each training round, each node receives data about its (in)direct neighbors (step ①), which is stored in its local database (step ②, Section V-C). This data, as well as the node's own model, is used by the aggregation function to produce the aggregated intermediary model (step ③, Section V-A). This intermediary model is trained on the node's local private dataset to produce the trained model (step ④), which serves as the node's own model during the next training round. Furthermore, for every neighbor, the node uses the probabilistic gossiping mechanism to select an entry from its local database (step ⑤, Section V-B). Both this probabilistically selected model and the node's own model are used in the composition of the broadcast message (step ⑥, Section V-D), which is transmitted to its neighbors.

### A. Aggregation function

We improve upon the intuitive direction of FoolsGold (Section III-A), designed for inherently unscalable federated learning. By exploiting the high degree of similarity between Sybil models, FoolsGold detects and diminishes the impact of Sybils on the training process. Based on this promising heuristic (Assumption 4), we adopt a modified version of FoolsGold as SybilWall's aggregation function (step ③). By doing so, SybilWall is able to directly mitigate the *dense Sybil poisoning attack*. In such a case, the aggregation function has knowledge of at least two directly connected Sybils producing highly similar models. These models are subsequently excluded during the aggregation as they are detected by the cosine similarity function. Furthermore, our aggregation function improves on FoolsGold in two dimensions.

Firstly, we modify FoolsGold to always trust the aggregator. Adversaries are not capable of compromising a node's training dataset and their training function by Assumption 1. Therefore, nodes can trust and exclude their own work from the similarity function during aggregation. The aggregator's model is reintroduced into the aggregation with the maximum weight, after all other models have been assigned a score.

Secondly, SybilWall supports incorporating additional data of indirect neighbors in the similarity function. This increases the probability of comparing the data of at least two Sybils, which do not necessarily have attack edges to the same honest node. By Assumption 4, these Sybils will produce similar

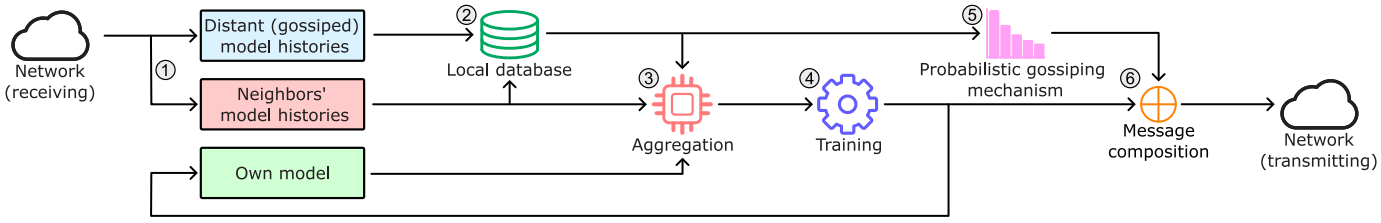



Figure 6: The Sybilwall architecture.

data each training round. Consequently, in the case where an honest node is only connected to a single attack edge, the incorporation of data from indirect neighbors might lead to the attack edge's mitigation. Although data from indirect neighbors are included in the similarity function, only data from direct neighbors are considered for aggregation. Note that obtaining data from indirect neighbors is not supported by the default decentralized learning API, but is facilitated through gossiping by the probabilistic gossiping mechanism.

### B. Probabilistic gossiping mechanism

We devised a probabilistic gossiping mechanism (step ⑤), which allows data dissemination among indirect neighbors. By doing so, the sensitivity of the aggregation function improves as the amount of reference material for the similarity function increases. Furthermore, this enables SybilWall to mitigate the *distributed Sybil poisoning attack*, as the probabilistic gossiping mechanism provides the aggregation function with gossiped data from a wider scope of nodes. The gossiped data consists of the model history $h_i^T$ of some node $i$ in round $T$, which is defined by $h_i^T = \sum_{t=0}^{T} w_i^t$, where $w_i^t$ is a trained model produced by node $i$ in round $t$.

First, let us define the method in which model histories are selected to be propagated to a neighboring node, for which SybilWall employs a weighted random selection algorithm.

More specifically, let $\mathcal{H}_i$ denote the local database of model histories of node $i$. $\mathcal{H}_i$ consists of a list of tuples, each in the form $\langle p, h, r, d, f \rangle \in \mathcal{H}_i$. Here, $h$ stands for the model history of node $p$ and $r$ signifies the identifier of the synchronous training round from which the model history originates. $d$ represents the distance that the model history has traveled, counted in the number of propagations. The term $f$ refers to the neighbor of node $i$ that provided this particular model history. Given the current node $i$ and its neighboring node $j$, let the filtered database of model histories $\mathcal{H}_i^j$ be defined as $H_i^j = \{(p, h, r, d, f) \in H_i \mid p \notin \{i, j\} \wedge f \neq j\}$. This filtered database is used in a weighted random selection to determine which model history will be gossiped to node $j$.

To perform the weighted random selection, the entries of the filtered database of model histories are first assigned weights. These weights directly correspond to the traveled distance $d$ and are assigned according to the exponential distribution:

$$P(d) = \lambda e^{-\lambda d} \tag{5}$$

where $\lambda$ can be considered a hyperparameter representing the relevance of propagating the model history of distant nodes. The choice for the exponential distribution is not arbitrary, as it prioritizes the propagation of the model history of nearby nodes over that of distant nodes. This approach assumes that the *sparse Sybil poisoning attack* is mitigated through a natural dampening effect. This natural dampening effect originates from the repeated train-aggregate loop (Figure 2) on each node, causing the influence of a Sybil to fade as the distance to the attack edge increases. After the weights have been assigned to the filtered database of model histories, a weighted random selection is performed to select the model history that is propagated.

### C. Local database updates

A node's local database of model histories can be updated in two distinct methods (step ②). First, if a node $i$ receives a model history through gossiping from some other node $j$, which it has not seen before, it is added to $i$'s local database. Second, if node $i$ receives a model history from some node $k$ that is more recent than the prior model history of $k$ known to node $i$, it is updated accordingly. Note that the model histories of direct neighbors are updated every round, as each training round will result in a more recent model history. It is possible that a node's local database of model histories may grow to a significant size over time, resulting in a decrease in performance during aggregation. In such a scenario, SybilWall can drop outdated model histories to prevent performance loss.

### D. Message composition

The previously described probabilistic gossiping mechanism requires model histories to be propagated to neighbors, which can be maliciously manipulated if implemented naively. As such, adversaries could exploit this by increasing the similarity of two targeted nodes, thereby potentially reducing the relative similarity among its Sybils. To mitigate this vulnerability, SybilWall replaces the original model communication discussed in Section II-B with a more secure communication scheme, which involves the use of signed histories.

To enable the use of signatures (secure by Assumption 2), the model history and the corresponding signature are constructed on the originating node. By doing so, any node can propagate signed model histories of (in)direct neighbors. However, this induces additional communication overhead since the trained model, the signed model history, and a signed gossiped model history all need to be communicated to neighbors every training round. We decrease these communication costs by

Table I: The default hyperparameters used during the evaluation of SybilWall.

| Hyperparameter | Value |
|---|---|
| # honest nodes | 99 |
| Attack edge density $\phi$ | 1 |
| Gossip mechanism parameter $\lambda$ | 0.8 |
| Dirichlet concentration parameter $\alpha$ | 0.1 |
| Max node degree $d$ | 8 |
| Local epochs | 10 |
| Batch size | 8 |

omitting the trained model, as it can be inferred from the comparison of the two most recent model histories.

More specifically, SybilWall composes messages (step ⑥) such that a message $m^T_{i\to j}$ from node $i$ to $j$ in round $T$ can be decomposed into $\langle M^T_i, M^r_k, d_{k\to i}+1\rangle$. In this decomposition, $M^T_i$ represents the model history information originating from node $i$ in round $T$, $M^r_k$ is the gossiped model history from node $k$ originating from round $r$ and $d_{k\to i}$ is the distance model history $M^r_k$ has traveled so far. $M^T_i$ can be further decomposed into $\langle h^T_i, i, T, S_i(h^T_i, T)\rangle$, where $h^T_i$ is the model history of node $i$ in round $T$ and $S_i$ is node $i$'s signature function. Note that all nodes construct their own model histories through cumulative summation of the trained model.

### E. Downtime tolerance

In contrast to a pull-based communication scheme, in which nodes could stochastically request a (distant) node's signed model history, SybilWall supports arbitrary downtime or the presence of private networks, both resulting in unreachable nodes. Moreover, a pull-based communication scheme would enable trivial manipulation of model histories, allowing adversaries to make Sybils seem more diverse. Using SybilWall's communication scheme, nodes are not responsible for the propagation of their own model history. Therefore, consistent reachability is not a necessity to proceed the training process.

In the event that a node experiences downtime, its aggregation function will start operating properly again once the node is online again and skips an additional training round. By doing so, it can obtain two subsequent model histories from its neighbors. This allows for inference of the trained model, since the difference between the two received model histories corresponds to the trained model required for aggregation.

## VI. Evaluation

We evaluate SybilWall by answering the following questions: *(1) How does the complexity of the dataset and the model affect the performance of SybilWall? (2) How does SybilWall perform compared to other existing algorithms? (3) How does the attack density $\phi$ influence the performance of SybilWall? (4) What is the effect of the distribution of data among nodes on the performance of SybilWall? (5) Can SybilWall be further enhanced by combining it with different techniques?*

Table II: The datasets used in the evaluation of SybilWall.

| Dataset | Model | Learning rate |
|---|---|---|
| MNIST [62] | Single soft-max layer | $\eta = 0.01$ [31] |
| FashionMNIST [49] | Single soft-max layer | $\eta = 0.01$ [31] |
| CIFAR-10 [47] | LeNet-5 [46] | $\eta = 0.004$ [63] |
| SVHN [64] | LeNet-5 [46] | $\eta = 0.004$ [63] |

### A. Experimental setup

We implemented SybilWall in Python3 in the context of a fully operational decentralized learning system for experimental evaluation and is online available [57]. We have used the PyTorch [58] library for the training of machine learning models. Regarding communication between individual nodes, we leveraged IPv8 [59], which provides an API for constructing network overlays in order to simulate P2P networks. Furthermore, we adopted the Gumby library [60] 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) [61]. Each node in the compute cluster has access to a dual 16-core CPU, 128 GB RAM, and either an A4000 or A5000 GPU. Furthermore, all default hyperparameters for the experiments can be found in Table I. Except where mentioned otherwise, these default hyperparameters define the configuration of all experiments.

In all experiments, we measure the accuracy by averaging the accuracy of the models of all honest nodes. Simultaneously, we measure the success rate of the attacker by averaging the attack score achieved on the models of all honest nodes. The attack score is defined as the accuracy that a model obtains on the altered segment of the data obtained by transforming the test dataset by the data transformation functions defined in Equation 3 and 4. Note that both metrics are measured each round directly after aggregation.

*1) Datasets:* The datasets used during evaluation can be found in Table II. These datasets were chosen for a number of reasons. First of all, MNIST [62] is a widely used dataset for the evaluation of machine learning algorithms [65]–[67], serving as an adequate baseline algorithm for SybilWall. FashionMNIST was developed as a more challenging variant of MNIST, thus serving as an ideal candidate to demonstrate the direct correlation between the complexity of classification tasks and the performance of SybilWall. The choice for SVHN and CIFAR-10 is motivated by the increased complexity of the models required to obtain satisfactory accuracy, which may affect the performance of SybilWall. The use of complex multilayer models in evaluation is frequently overlooked in related work or is performed only on a single dataset [65]–[69]. Moreover, when multilayer models are used, they are regularly pre-trained and trained solely through transfer learning [67], [70]. While we recognize that all the datasets employed in this experimental evaluation focus on image classification, we argue 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 distributed machine learning

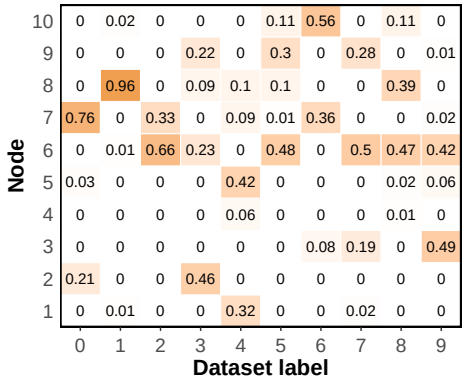


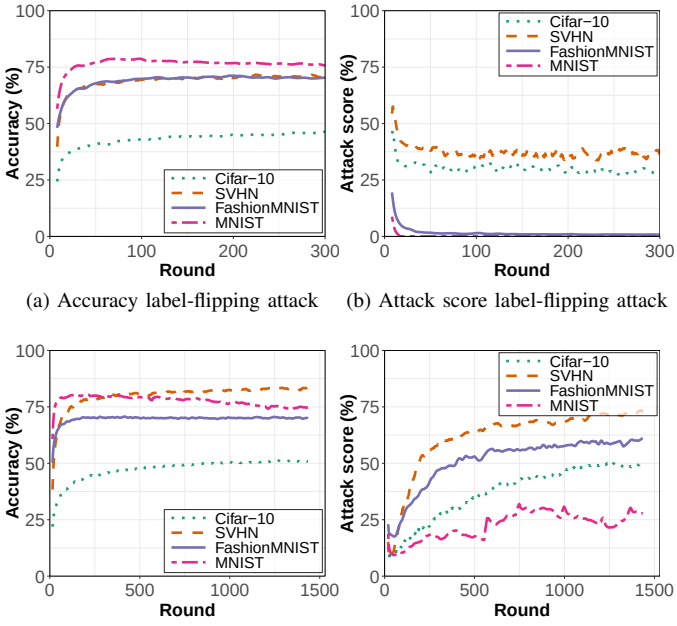

Figure 7: Example distribution for non-i.i.d. data generated with the Dirichlet distribution with concentration parameter $\alpha = 0.1$ for 10 nodes and a dataset containing 10 labels.

algorithms [27], [65]–[69], and there exists a variety of widely available datasets constructed specifically for this task.

The models that are trained using the aforementioned datasets can also be found in Table II, as well as the corresponding learning rate $\eta$. Note that all the models in this evaluation are neural networks and are trained using stochastic gradient descent (SGD).

*2) Data distribution:* The aforementioned datasets are designed for centralized machine learning and require to be distributed among the participating nodes. During our evaluations, we assume that the data is *not* identically and independently distributed (non-i.i.d.), which more closely resembles real-world data than uniformly distributed data (i.i.d) [71], [72]. Some studies employ the use of a K-shard data distribution [9], [67], [73], [74] or simply assign each node a predefined number of classes of the training data [73], [75]. However, we utilize the Dirichlet distribution [76], which has recently gained more popularity for generating non-i.i.d. data distributions [27], [77], [78]. More specifically, given the *concentration parameter* $\alpha$, we compute for each class the fraction of data every node possesses using the Dirichlet distribution, creating seemingly naturally unfair and irregular data distributions. Lower values of $\alpha$ result in more non-i.i.d. data. Figure 7 illustrates an example distribution for a dataset of 10 labels distributed over 10 nodes with a concentration parameter of $\alpha = 0.1$.

*3) Network topology:* To generate the necessary network topologies, defining the relations between nodes, we employed *random geometric graphs*. Random geometric graphs are constructed by randomly placing points, which correspond to nodes, on a grid. Two nodes are connected by an edge when the Euclidean distance between the corresponding points of these nodes is smaller than some predefined constant. To enforce the upper bound on a node's degree (Assumption 6), random edges are removed from the random geometric graph, such that all nodes remain connected through a single connected component. The code used to generate these network topologies can be found in our published code repository




(a) Accuracy label-flipping attack (b) Attack score label-flipping attack

(c) Accuracy backdoor attack (d) Attack score backdoor attack

Figure 8: Accuracy and attack score for the label-flipping attack (300 rounds) and the backdoor attack (1450 rounds) on different datasets.

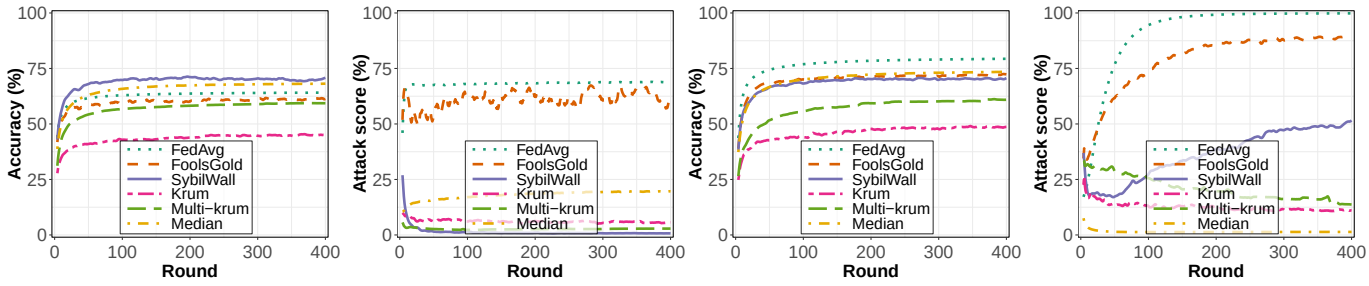
[57]. Furthermore, during our experiments, we assume a static network topology. That is, no nodes will leave or join the network during training, including Sybils. Lastly, we employ the SSP attack (Section IV-C) as the adversarial strategy in the simulated Sybil attacks, since we hypothesize that more distant attack edges will result in a lower detection rate, thereby approximating the optimal attack scenario.

### B. Effect of dataset

*1) Setup:* We evaluated the performance of SybilWall on different datasets, allowing us to observe how SybilWall is affected by varying the complexity in both the dataset and the model. This experiment was carried out using the default parameters listed in Table I and using the datasets, models and learning rates listed in Table II.

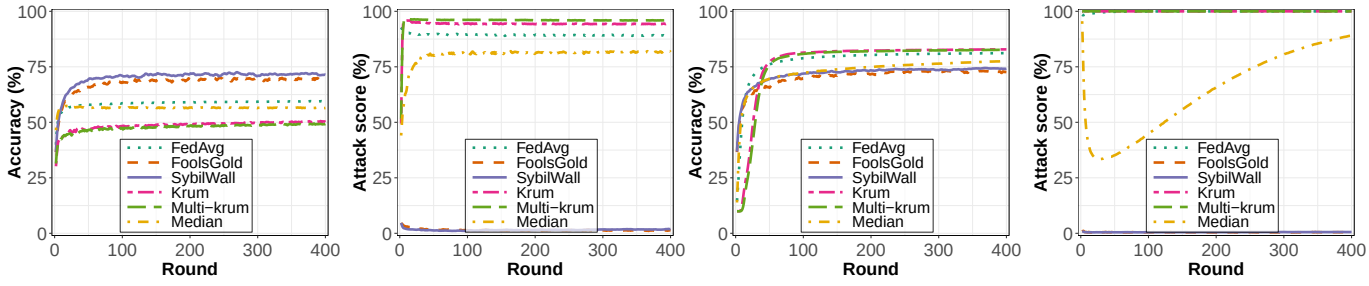
*2) Results:* Figure 8 demonstrates the effect of varying the dataset on the trend of accuracy and attack score. We clearly observe that CIFAR-10, arguably the most challenging dataset used in this work, obtains a significantly lower accuracy compared to simpler datasets (Figure 8a), such as MNIST. This can be explained by difference in the complexity of the training samples, i.e. MNIST consists of grayscale images, while CIFAR-10 has RGB images. Moreover, samples in the CIFAR-10 dataset have more variety within a class, such as different backgrounds or different races of dogs.

A noteworthy observation with regard to the attack score of the label-flipping attack in Figure 8b is that datasets that require more sophisticated models, such as LeNet-5, are generally more susceptible to the label-flipping attack




(a) Accuracy label-flipping attack (b) Attack score label-flipping attack (c) Accuracy backdoor attack (d) Attack score backdoor attack

Figure 9: Comparison of SybilWall against different techniques on $\phi = 1$. Results generated using the FashionMNIST [49] dataset.




(a) Accuracy label-flipping attack (b) Attack score label-flipping attack (c) Accuracy backdoor attack (d) Attack score backdoor attack

Figure 10: Comparison of SybilWall against different techniques on $\phi = 4$. Results generated using the FashionMNIST [49] dataset.

compared to simpler models, such as a single-layer neural network. The results suggest that the smaller number of trainable weights in the single-layer neural network cause the Sybil model histories to exhibit greater similarity relative to more sophisticated models when comparing model histories from different training rounds. Moreover, the greatly increased number of weights of the more sophisticated models allows for more diversity in the sum of the trained models.

Taking into account the results of the backdoor attack depicted in Figures 8c and 8d, it is apparent that all attack scores demonstrate an increasing trend over a prolonged period of time. However, if we consider the convergence rate of both the accuracy and attack score, it becomes apparent that the attack score requires a substantially longer time period to reach convergence on most datasets.

### C. Comparison with different techniques

*1) Setup:* We evaluate the performance of SybilWall relative to a number of different techniques focused on mitigating Sybil poisoning attacks or Byzantine attacks in general. These techniques are the following:

i. FedAvg [9]: naively averages all models. This algorithm was the first proposed federated learning aggregation algorithm and will serve as a baseline during our evaluation.
ii. FoolsGold [31]: detects Sybils among its neighbors by assuming that Sybils produce highly similar models.
iii. Krum [48]: excludes Byzantine models by filtering for the model which has the smallest sum of Euclidean distances to its $n - f - 2$ closest neighbors.
iv. Multi-krum [48]: similar to krum. Averages the $m$ models with the lowest sum of Euclidian distances to its $n-f-2$ closest neighbors.
v. Median [79]: computes the element-wise median of all models and thereby excludes outliers.

During this experiment, we alternated the attack edge density $\phi \in \{1, 4\}$ and fixed the dataset on FashionMNIST.

*2) Results:* Figure 9 shows the results of SybilWall compared to different techniques using attack edge density $\phi = 1$. We observe that SybilWall always scores among the best performing algorithms in terms of accuracy. Especially considering the label-flipping attack, SybilWall achieves the highest accuracy among all evaluated techniques. Furthermore, the results demonstrate that SybilWall successfully mitigates the label-flipping attack, similarly to some of the other techniques evaluated. On the backdoor attack, we observe that SybilWall exhibits the same increasing trend as in the prior experiment on the effect of the datasets in Section VI-B; the initially low attack score gradually increases as training progresses.

Figure 10 shows the results of SybilWall compared to different techniques using a higher attack edge density $\phi = 4$. These results clearly demonstrate how most aggregation algorithms succumb under the use of a large-scale Sybil attack. Taking into account the accuracy of both label-flipping attack and

backdoor attack, we observe that the converged accuracy of most algorithms increases significantly when employing the backdoor attack. This phenomenon can be explained by the fact that the adversary is not actively attempting to decrease the accuracy of the model. In fact, the adversary only attempts to insert an activation pattern, which was highly successful for the algorithms demonstrating an increased converged accuracy compared to the label-flipping attack. On the other hand, both FoolsGold and SybilWall seem to be unaffected by both attacks. Regarding SybilWall, this is likely caused by the integration of a modified version of FoolsGold, which was designed to mitigate the *dense Sybil poisoning attack*.

Considering both the results in Figure 9 and 10, Krum and Multi-krum algorithms surprisingly obtain a higher accuracy under a backdoor attack with higher attack edge density compared to a lower attack edge density, while the accuracy of the other algorithms remain relatively constant in both scenarios. Lastly, we find that SybilWall does not outperform all the alternative evaluated algorithms in all scenarios, but it is the only algorithm to consistently score among the best.

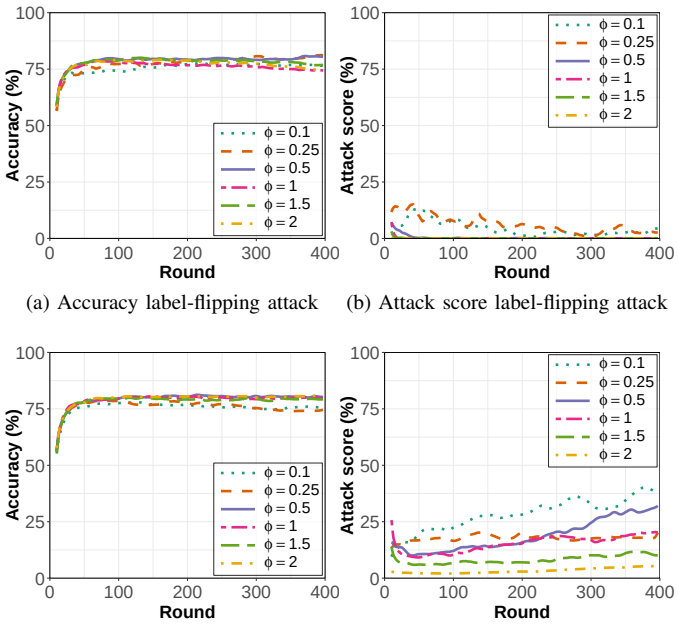
### D. Effect of attack edge density

*1) Setup:* We evaluate SybilWall in a number of different attack edge density configurations. This experiment aims to demonstrate the effect that an attacker can exercise on the network by creating different numbers of Sybils. MNIST is fixed as the dataset during this experiment and the attack edge density $\phi$ is varied within the range $\phi \in [0.1, 2]$.

*2) Results:* Figure 11 illustrates the effect of various attack edge density values on the label-flipping attack and backdoor attack. It is apparent that the attack edge density has little effect on the converged accuracy (Figures 11a and 11c). However, Figure 11d shows how the attack score decreases as the attack edge density increases. This suggests that SybilWall successfully decreases the utility gained from creating additional Sybils, as it will likely decrease the attack score. A similar effect can be observed with the label-flipping attack (Figure 11b), where a lower attack edge density results in higher attack score. The attack score of the label-flipping attack also demonstrates how SybilWall becomes increasingly stronger in reducing the attack score over time. This can be explained by the probabilistic gossiping mechanism causing knowledge about distant nodes to accumulate over time, thereby improving the ability to detect Sybils among direct neighbors.

### E. Effect of data distribution

*1) Setup:* The method in which the data is distributed over the nodes might influence the attack score and the accuracy of the trained models. To explore this effect, we evaluate SybilWall's performance under a variety of data distributions. More specifically, we vary the data distribution between i.i.d. and non-i.i.d. (Dirchlet-based). For the non-i.i.d. scenario, we vary the concentration parameter $\alpha$ within the range $\alpha \in [0.05, 1]$. Furthermore, we fixate the dataset on CIFAR-10.




(a) Accuracy label-flipping attack (b) Attack score label-flipping attack

(c) Accuracy backdoor attack (d) Attack score backdoor attack

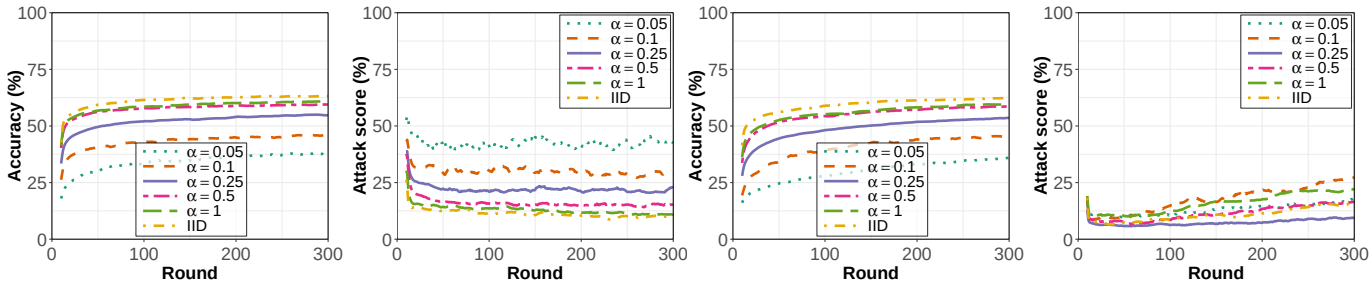
Figure 11: Accuracy and attack score for the label-flipping attack and backdoor attack on different attack edge densities. Results generated using the MNIST [62] dataset.

*2) Results:* Figure 12 shows the effects of different data distributions on the convergence of the training process. We observe in both the label-flipping attack and backdoor attack that the accuracy increases as the data is more uniformly distributed (Figures 12a and 12c). Furthermore, the attack score of the label-flipping attack demonstrates how the attack score decreases as the training data are more uniformly distributed (Figure 12b). This finding suggests that adversaries will be more successful in networks with highly non-i.i.d. data. This is likely due to the decrease in the number of nodes capable of counteracting the label-flipping attack, as they are less likely to possess training samples belonging to the targeted classes. Lastly, the data distribution does not appear to have a significant effect on the attack score of the backdoor attack, as no clear trend emerges when varying the data distribution (Figure 12d). This can be explained by the fact that counteracting the backdoor attack does not require the possession of specific training data.

### F. Further enhancing SybilWall

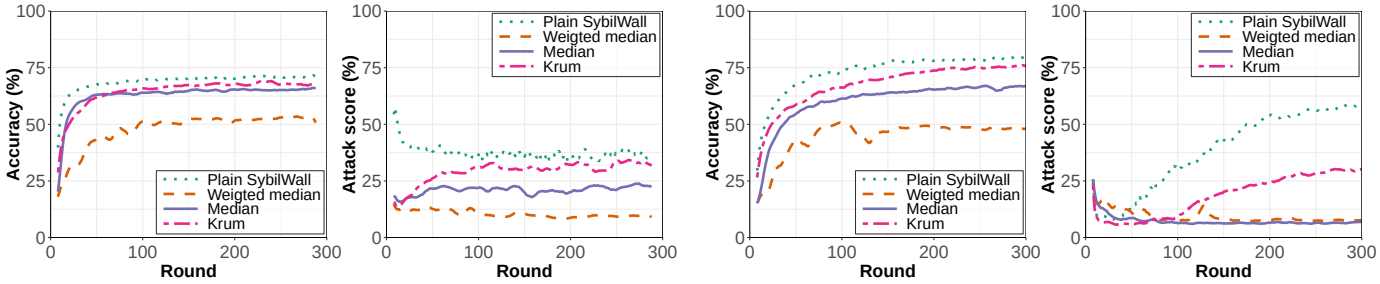
*1) Setup:* Given the increasing, although impeded, attack score demonstrated for the backdoor attack in Section VI-B, we consider several techniques for additional enhancement of the defensive capabilities of SybilWall. These augmentations include the following:

i. Median: given the resilience of the Median [79] algorithm in Section VI-C against attack edge density $\phi = 1$, we implement a combined version of the Median approach and SybilWall. This is achieved by initially employ-




(a) Accuracy label-flipping attack (b) Attack score label-flipping attack (c) Accuracy backdoor attack (d) Attack score backdoor attack

Figure 12: Accuracy and attack score for the label-flipping attack and backdoor attack of different data distributions, indicated by the concentration parameter $\alpha$ of the Dirichlet distribution. Results generated using the CIFAR-10 dataset [47].




(a) Accuracy label-flipping attack (b) Attack score label-flipping attack (c) Accuracy backdoor attack (d) Attack score backdoor attack

Figure 13: Accuracy and attack score of the label-flipping attack and backdoor attack for different possible enhancements of SybilWall. Results generated using the SVHN [64] dataset.

ing SybilWall to compute a non-normalized aggregation weight in the range $[0, 1]$, followed by the execution of the Median algorithm on the 50% highest scoring models.

ii. Weighted median: a variant of the Median-based approach, in which scores computed by SybilWall are adopted as weights for a weighted median aggregation.

iii. Krum-filter: based on the suggestion of prior work [31], we combine SybilWall with Krum, such that the model with the lowest Krum score receives an aggregation weight of 0.

We integrate these augmentations through chaining the aggregation functions, such that the last step of SybilWall's aggregation method, a weighted average, is substituted. We also provide the trends for plain SybilWall to serve as a baseline. The dataset is fixed to SVHN.

*2) Results:* Figure 13 illustrates the effect of enhancing SybilWall with various methodologies. We find that plain SybilWall achieves the highest accuracy overall, but the worst Sybil resilience. While each of the evaluated methodologies improves SybilWall's defensive capabilities, a trade-off occurs in which accuracy is sacrificed to obtain a decreased attack score. In particular, the low attack score of the weighted median is unmatched, but achieves considerably lower accuracy compared to the alternative methodologies. The Krum-filter-based approach appears to obtain an accuracy comparable to plain SybilWall, but it obtains the worst Sybil resilience of the evaluated enhancements. However, it exhibits a substantially lower attack score compared to plain SybilWall. Arguably, the Median-based methodology shows the most promising results, as it achieves to consistently limit the attack score to levels comparable to those of the weighted median methodology, while maintaining a significantly higher accuracy.

## VII. Discussion

During the experimental evaluation of SybilWall, we performed various experiments to assess the performance and Sybil poisoning resilience of SybilWall. First, we measured the performance of SybilWall against 4 widely adopted datasets. We argue that SybilWall obtained satisfactory accuracy and convergence rate on all of these datasets. Furthermore, the converged accuracy obtained by SybilWall is similar to that achieved by the FedAvg algorithm in a federated learning setting (Figures 3 and 8). In addition to obtaining satisfactory accuracy on all datasets, we also compared SybilWall against a number of alternative algorithms and found that SybilWall was the only evaluated algorithm that consistently scored among the best algorithms in all scenarios. SybilWall therefore arguably exhibits the overall strongest resilience to Sybil poisoning attacks and possesses the qualities to be considered state-of-the-art. Although the attack score of the backdoor attack shows a rising trend when employing SybilWall, we note that the convergence rate is greatly reduced compared to alternative algorithms. This allows honest nodes to stop the training process once the accuracy has converged, thus limiting the success of potential adversaries.

We argue that the aforementioned rising trend demonstrated by the attack score of the backdoor attack mainly originates from the difficulty of counteracting the effect of the backdoor attack, as no node will possess training samples directed to mitigate the specific activation pattern. Therefore, SybilWall's only method of mitigating the backdoor attack is by detecting highly similar behavior from Sybils. One might argue that the summation of a node's trained models may not accurately reflect the node's behavioral history, as it is highly affected by the aggregated intermediary model. Similarly to prior work [31], summing a model's trained gradients, rather than the model itself, would arguably improve the representation of a node's behavioral history. Using this approach, a node's history would more closely correspond to a node's training data, thereby decreasing the influence of the aggregated intermediary model. Furthermore, it would also better represent how a node aims to contribute to the aggregated model.

Assumption 5 implies that all Sybils will distribute the same model history each training round, as the adversary does not have sufficient computational power to train multiple aggregated intermediary models. This assumption was made to support Assumption 4, since Sybils would likely produce highly diverse model histories if each Sybil had a unique aggregated intermediary model. An example of such a situation is click farms [80], which provide an adversary with extensive computation capabilities, violating Assumption 5. If one were to adopt the use of gradient history rather than model history, the Sybils might exhibit highly similar behavior regardless of the aggregated intermediary model. As Sybils would share the same altered training data, we argue that their summed model gradients will demonstrate more similarity than those of honest nodes. This improvement might eventually lead to the omission of Assumption 5.

On the contrary, obtaining a node's model gradients is a non-trivial task in the setting of decentralized learning. It would be highly challenging to verify the validity of the aggregated intermediary model on which the gradients were obtained. This aggregated intermediary model would also need to be communicated to allow neighbors to construct the trained model. One option to verify its validity is by sharing the private training data and violating the user's privacy. However, respecting the user's privacy is one of the fundamental arguments for federated and decentralized learning [9]. On the other hand, if one were to allow the usage of unverified aggregated intermediary models, adversaries could trivially launch Sybil poisoning attacks with highly diverse behavior. As an example of such an attack, a Sybil could first train the aggregated intermediary model on the altered training dataset to obtain a malicious trained model $m_s$. Secondly, the Sybil can generate an arbitrary aggregated intermediary model $m_r$ such that the gradients $g$ are defined as $g = m_s - m_r$. Since the validity of the arbitrary aggregated intermediary model $m_r$ cannot be validated, each Sybil could create highly diverse model gradients. This attack is not possible in federated learning, as the aggregated intermediary model is equal for all nodes every round and was created by a central authority.

As previously mentioned, adopting the usage of gradient histories requires the ability to verify the validity of the aggregated intermediary model. Mao et al. [81] suggests a method which achieves this by reaching a global consensus on the aggregated intermediary model. By repeatedly averaging the model with that of neighbors, nodes converge to a globally coherent model under the assumption that every node will participate honestly while attempting to reach consensus. However, we argue that this assumption does not realistically reflect a deployed decentralized setting and is therefore not applicable to this study. We leave the required analysis for a robust method of verifying the validity of the aggregated intermediary model to future work.

During the evaluation of the effect of varying the attack edge density on the attack score in Section VI-D, we found that decreasing the number of Sybils increases the attack score. Therefore, we hypothesize that SybilWall eliminates the need to amplify a poisoning attack with the Sybil attack, as employing Sybils would result in a lower attack score. However, SybilWall does not have the ability to successfully mitigate a single-attacker poisoning attack. Mitigating such an attack would require the incorporation of an alternative poisoning attack mitigation algorithm. Section VI-F explores further enhancing SybilWall with a number of such alternative algorithms. Although all enhancements demonstrated increased resilience to Sybil poisoning, they all sacrifice in terms of accuracy. Considering that accuracy is often the primary goal in machine learning [82], the justification of such enhancements is highly dependent on the application and its users. We leave further enhancing SybilWall for increased single-attacker poisoning mitigation as a possible research direction for future work. Ideally, this algorithm would increase the resilience against individual attack edges without compromising accuracy.

Furthermore, adversaries may employ a strategy to generate more diverse Sybil model histories. By introducing random noise to the irrelevant weights of the model [31], adversaries may be able to significantly increase the diversity among Sybil model histories, resulting in a violation of Assumption 4. More research is required to accurately filter for relevant weights, which could be achieved through a number of approaches, such as layer-wise relevance propagation [83], weight magnitude filtering [84], or empirical weight importance [85].

## VIII. Conclusion

We have presented SybilWall, a pioneering algorithm in the mitigation of Sybil poisoning attacks in decentralized learning. Building on the Sybil poisoning mitigation algorithm, FoolsGold [31] (federated learning), we exploit the increased similarity between the models produced by Sybils over that of honest nodes. We proposed a probabilistic gossiping mechanism to facilitate data dissemination. The disseminated data aids in the mitigation of a poisoning attack amplified by distributing Sybils over the decentralized network. We argue that SybilWall achieves satisfactory performance on four widely adopted datasets and obtains similar accuracy

to federated learning. Furthermore, we empirically evaluated SybilWall against several alternative algorithms. Our findings indicate SybilWall to be the only algorithm that consistently scored among the best in all evaluated scenarios, thus arguably outperforming all alternative evaluated algorithms. Although SybilWall does not fully mitigate targeted poisoning attacks in the form of a backdoor attack, it manages to greatly decrease the convergence rate of the attacker's success. This enables honest nodes to complete the training process prior to the attack having substantial impact.

We proposed a number of promising future research directions, such as further improving SybilWall to deflect single attackers, or exploring potential improvements to better mitigate the backdoor attack by adopting the usage of summed model gradients in the similarity metric.

## 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] 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.

[3] 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

[4] 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

[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] R. Shao, H. He, H. Liu, and D. Liu, "Stochastic channel-based federated learning for medical data privacy preserving," 2019.

[8] 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.

[9] 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

[10] 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.

[11] 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

[12] 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

[13] 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

[14] 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

[15] 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.

[16] 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

[17] 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.

[18] 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

[19] 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

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

[21] 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

[22] 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

[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] 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

[25] 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

[26] 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.

[27] 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.

[28] 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

[29] 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.

[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] 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.

[33] 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

[34] 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

[35] 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.

[36] 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

[37] 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

[38] 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

[39] ——, "Mitigating backdoor attacks in federated learning," *CoRR*, vol. abs/2011.01767, 2020. [Online]. Available: https://arxiv.org/abs/2011.01767

[40] 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.

[41] 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.

[42] H. Rowaihy, W. Enck, P. McDaniel, and T. La Porta, "Limiting sybil attacks in structured p2p networks," in *IEEE INFOCOM 2007 - 26th IEEE International Conference on Computer Communications*, 2007, pp. 2596–2600.

[43] Y. Xie, F. Yu, Q. Ke, M. Abadi, E. Gillum, K. Vitaldevaria, J. Walter, J. Huang, and Z. M. Mao, "Innocent by association: Early recognition of legitimate users," in *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, ser. CCS '12. New York, NY, USA: Association for Computing Machinery, 2012, p. 353–364. [Online]. Available: https://doi.org/10.1145/2382196.2382235

[44] F. Lesueur, L. Mé, and V. V. T. Tong, "A sybil-resistant admission control coupling sybilguard with distributed certification," in *2008 IEEE 17th Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises*, 2008, pp. 105–110.

[45] M. Moradi and M. Keyvanpour, "Captcha and its alternatives: A review," *Security and Communication Networks*, vol. 8, no. 12, pp. 2135–2156, 2015. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/sec.1157

[46] 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.

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

[48] 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.

[49] 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

[50] M. de Vos and J. Pouwelse, "Contrib: Maintaining fairness in decentralized big tech alternatives by accounting work," *Computer Networks*, vol. 192, p. 108081, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128621001705

[51] Q. Stokkink, C. U. Ileri, D. Epema, and J. Pouwelse, "Web3 sybil avoidance using network latency," *Computer Networks*, vol. 227, p. 109701, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128623001469

[52] M. Gjoka, M. Kurant, C. T. Butts, and A. Markopoulou, "Walking in facebook: A case study of unbiased sampling of osns," in *2010 Proceedings IEEE INFOCOM*, 2010, pp. 1–9.

[53] A. Singh, T.-W. Ngan, P. Druschel, and D. S. Wallach, "Eclipse attacks on overlay networks: Threats and defenses," in *Proceedings IEEE INFOCOM 2006. 25TH IEEE International Conference on Computer Communications*, 2006, pp. 1–12.

[54] 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.

[55] 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

[56] L. Kaufman and P. J. Rousseeuw, *Finding groups in data: an introduction to cluster analysis*. John Wiley & Sons, 2009.

[57] T. Werthenbach and J. Pouwelse, "Towards sybil resilience in decentralized learning," https://doi.org/10.5281/zenodo.8077387, Jun 2023.

[58] 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

[59] Tribler, "Python implementation of tribler's ipv8 p2p-networking layer," https://github.com/Tribler/py-ipv8, 2023.

[60] ——, "Experiment runner framework for ipv8 and tribler," https://github.com/Tribler/gumby, 2022.

[61] 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.

[62] 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.

[63] 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

[64] 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

[65] C. Pappas, D. Chatzopoulos, S. Lalis, and M. Vavalis, "Ipls: A framework for decentralized federated learning," in *2021 IFIP Networking Conference (IFIP Networking)*, 2021, pp. 1–6.

[66] S. Alqahtani and M. Demirbas, "Performance analysis and comparison of distributed machine learning systems," *CoRR*, vol. abs/1909.02061, 2019. [Online]. Available: http://arxiv.org/abs/1909.02061

[67] J. Verbraeken, M. de Vos, and J. Pouwelse, "Bristle: Decentralized federated learning in byzantine, non-i.i.d. environments," *CoRR*, vol. abs/2110.11006, 2021. [Online]. Available: https://arxiv.org/abs/2110.11006

[68] H. Ye, L. Liang, and G. Y. Li, "Decentralized federated learning with unreliable communications," *IEEE Journal of Selected Topics in Signal Processing*, vol. 16, no. 3, pp. 487–500, 2022.

[69] 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

[70] K. Weiss, T. M. Khoshgoftaar, and D. Wang, "A survey of transfer learning," *Journal of Big data*, vol. 3, no. 1, pp. 1–40, 2016.

[71] T.-C. Chiu, Y.-Y. Shih, A.-C. Pang, C.-S. Wang, W. Weng, and C.-T. Chou, "Semisupervised distributed learning with non-iid data for aiot service platform," *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 9266–9277, 2020.

[72] K. Hsieh, A. Phanishayee, O. Mutlu, and P. Gibbons, "The non-IID data quagmire of decentralized machine 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. 119. PMLR, 13–18 Jul 2020, pp. 4387–4398. [Online]. Available: https://proceedings.mlr.press/v119/hsieh20a.html

[73] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-iid data," *CoRR*, vol. abs/1806.00582, 2018. [Online]. Available: http://arxiv.org/abs/1806.00582

[74] Y. Chen, Y. Ning, M. Slawski, and H. Rangwala, "Asynchronous online federated learning for edge devices with non-iid data," in *2020 IEEE International Conference on Big Data (Big Data)*, 2020, pp. 15–24.

[75] C. Briggs, Z. Fan, and P. Andras, "Federated learning with hierarchical clustering of local updates to improve training on non-iid data," in *2020 International Joint Conference on Neural Networks (IJCNN)*, 2020, pp. 1–9.

[76] G. L. Dirichlet, "Über die reduction der positiven quadratischen formen mit drei unbestimmten ganzen zahlen." *Journal für die reine und angewandte Mathematik (Crelles Journal)*, vol. 1850, no. 40, pp. 209–227, 1850. [Online]. Available: https://doi.org/10.1515/crll.1850.40.209

[77] L. Gao, H. Fu, L. Li, Y. Chen, M. Xu, and C.-Z. Xu, "Feddc: Federated learning with non-iid data via local drift decoupling and correction," 2022.

[78] X. Mu, Y. Shen, K. Cheng, X. Geng, J. Fu, T. Zhang, and Z. Zhang, "Fedproc: Prototypical contrastive federated learning on non-iid data," *Future Generation Computer Systems*, vol. 143, pp. 93–104, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X23000262

[79] D. Yin, Y. Chen, K. Ramchandran, and P. L. Bartlett, "Byzantine-robust distributed learning: Towards optimal statistical rates," *CoRR*, vol. abs/1803.01498, 2018. [Online]. Available: http://arxiv.org/abs/1803.01498

[80] E. Drott, "Fake streams, listening bots, and click farms: Counterfeiting attention in the streaming music economy," *American Music*, vol. 38, no. 2, pp. 153–175, 2020.

[81] 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.

[82] S. Kaur and S. Jindal, "A survey on machine learning algorithms," *Int J Innovative Res Adv Eng (IJIRAE)*, vol. 3, no. 11, pp. 2349–2763, 2016.

[83] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, and W. Samek, "On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation," *PLOS ONE*, vol. 10, no. 7, pp. 1–46, 07 2015. [Online]. Available: https://doi.org/10.1371/journal.pone.0130140

[84] M. C. Mozer and P. Smolensky, "Skeletonization: A technique for trimming the fat from a network via relevance assessment," in *Advances in Neural Information Processing Systems*, D. Touretzky, Ed., vol. 1. Morgan-Kaufmann, 1988.

[85] J.-H. Luo, J. Wu, and W. Lin, "Thinet: A filter level pruning method for deep neural network compression," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.