Towards Sybil Resilience in Decentralized Learning

— MSc. Thesis —

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

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

***Abstract*—Decentralized learning has recently been emerging as a promising alternative to the privacy-enforcing distributed machine learning technology, federated learning. The scalability of federated learning is limited by the internet connection and memory capacity of the central parameter server, and the (non-linear) aggregation function. Decentralized learning obviates the need of a central parameter server by decentralizing the aggregation process 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 poses as the main motivator for this work, in which we aim to study and suggest a novel solution for improving the Sybil poisoning resilience of decentralized learning.**

We present SybilWall, a pioneering algorithm focused on increasing the resilience of decentralized learning against targeted Sybil poisoning attacks. By combining a modified version of FoolsGold [1], a similarity-based Sybil-resistant aggregation function designed for federated learning, with a novel probabilistic gossiping mechanism, we set a new baseline for Sybil resilient decentralized learning.

We performed extensive empirical evaluations of SybilWall, and found that SybilWall significantly outperforms existing state-of-the-art solutions designed for federated learning scenarios, and is the only aggregation function to achieve consistent accuracy over numerous adversarial strategies. Furthermore, SybilWall eliminates the need of spawning additional Sybils, as our evaluations indicate a higher success rate amongst adversaries employing a smaller number of Sybils. We conclude this work by discussing possible improvements of SybilWall and highlighting 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 [2], recommendation in social media [3] and music [4], and cybersecurity [5]. However, accurate machine learning models require large training datasets [6], [7], which can often be hard to obtain and store due to recent privacy legislation [8]. Federated learning [9] has become a promising alternative and widely adopted tool for crowd sourcing computationally expensive machine learning operations, reportedly having been used for training numerous industrial machine learning models [10]–[14]. Federated learning ensures the protection of privacy, as the user's data will not leave their device during training.

With federated learning, in contrast to centralized machine learning, training takes place on the end-users' personal devices, which are often referred to *edge devices* or *nodes*. The resulting trained models are communicated to a central parameter server, commonly referred to as the *parameter server*, which aggregates these models using some predefined methodology. By only sharing the end user-trained models with the parameter server, the user's privacy is preserved, while obtaining comparable performance compared to centralized machine learning [15]. While there exist attacks in which training data can be reconstructed based on the gradient of the trained models [16], [17], defense mechanisms against this attack have been proposed [18], [19].

However, federated learning suffers from some disadvantages. For instance, the parameter server aggregates the models of all participating nodes, inducing heavy communication costs and a potential bottleneck in the learning process affecting the overall convergence time [20]. Secondly, the scalability in terms of the amount of nodes heavily varies depending on the aggregation method. In secure and robust federated learning aggregation methods, the incorporation of additional nodes during aggregation may result in significantly increased computational effort for the parameter server [21]. Thirdly, the parameter server performing the aggregation poses a single-point of failure [22]. Disruptions to the parameter server can cause downtime and hinder the overall model training process, particularly in architectures where nodes require the globally aggregated model before continuing their training. An upcoming alternative aiming to resolve these issues is *decentralized learning*, also commonly referred to as *decentralized federated learning*. In decentralized learning, there exists no dedicated parameter server performing the aggregation and the nodes form a distributed network, e.g. a peer-to-peer network, in which each node individually performs aggregation on their neighbours' models (see Figure 1). While the information available during aggregation is more limited relative to federated learning, it has been shown that decentralized learning has the potential to obtain similar results compared to federated learning [23]. Models are exchanged between individual devices and aggregated on individual scale using some predefined aggregation method, alleviating the commu-

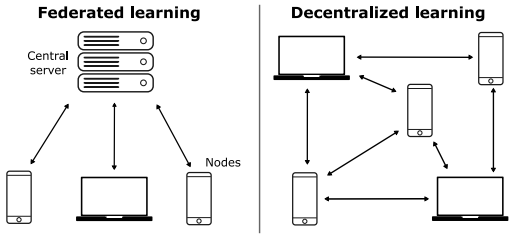


Figure 1: Federated learning compared to decentralized learning. Arrows represent a connection between two nodes and indicates the two connecting nodes share model updates during each training round.

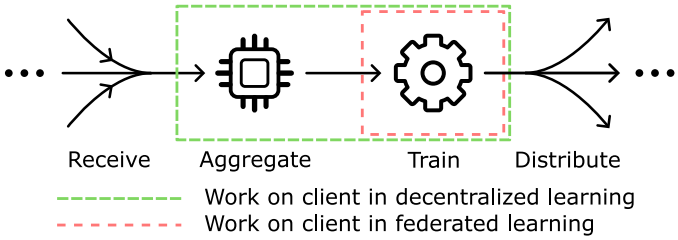


Figure 2: The general train-aggregate loop executed by all nodes participating in decentralized learning. Note the distinction in the work performed by participating nodes in federated learning and decentralized learning.

nicative bottleneck and single point of failure issues imposed on federated learning, and paving the path for boundless scalability.

While decentralized learning solves the scalability challenges faced in federated learning, it is still vulnerable to byzantine environments [24]. Since the predefined aggregation method in decentralized learning does not have access to all models in the network, aggregation is performed with less information compared to federated learning, resulting in relatively less resistance against possible poisoning attacks [25]. Poisoning attacks are can generally be categorized in two categories, namely those of *targeted poisoning attacks* and *untargeted poisoning attacks*. Targeted poisoning attacks focus on achieving a specific goal an adversary aims to achieve such as the label-flipping attack [26], [27] and the backdoor attack [28]–[30]. On the other hand, untargeted poisoning attacks aim to hinder the result of the training process in some way without any particular goal in mind. The effect of these attacks can often be amplified through combining them with the Sybil attack [31], in which an adversary controls a substantial amount of nodes to increase its influence. As such, an adversary may deploy the Sybil attack to rapidly spread their poisoned model through the network. In this work, we focus exclusively on targeted poisoning attacks amplified by Sybil attacks in decentralized learning.

Prior work on resilience against poisoning attacks combined with Sybil attacks in distributed machine learning has mainly been done in federated learning settings. One popular example of such work is *FoolsGold* [1], which aims to increase Sybil resilience under the assumption that all Sybils will broadcast similar gradients during each round of training. By dynamically adapting the aggregation weights of peers' models based on their similarity with others, experimental results suggest that FoolsGold has the potential to provide effective protection against Sybil attacks in small-scale and simple federated learning settings.

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

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

- We evaluate FoolsGold, a popular Sybil resilience algorithm in federated learning, and assess its compatibility with decentralized learning in Section III.
- We define the Spread Sybil Poisoning attack for effective Sybil poisoning attacks in decentralized learning, and decompose it into three distinct adversarial scenarios.
- We present SybilWall, a pioneering algorithm for Sybil resilience with boundless scalability in decentralized learning, in Section V.
- We perform an empirical evaluation of SybilWall's performance in VI on several datasets and against competitive alternatives.

II. Background

A. Federated learning

Federated learning was initially proposed by Google [9] as a means for training machine learning models on real user data without compromising users' privacy. This is achieved by training the machine learning model on the edge devices, which contain the real user data. The training proceeds in synchronous rounds, each consisting of a predefined number of epochs, during which the trained models are sent to the parameter server at the end of each round. The role of the parameter server is to aggregate all trained models into a global model without the need of any training data. After the models are aggregated, the global model is communicated back to the edge devices, after which the next training round commences. See Figure 1 for a simplified federated learning network topology. The original paper [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 is of the training process is to minimize the global loss function such that the global model x approaches the optimal model x^*. More formally, the search for a global optimal model can approximately be defined as:

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

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

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

B. Decentralized learning

Decentralized learning is an upcoming alternative for federated learning [33]–[36]. In contrast to federated learning, which relies on a parameter server for aggregating locally trained models, aggregation in decentralized learning takes place at a smaller scale and is performed by every participating node on their own model and those of its neighbours (see Figure 1 and 2). By doing so, numerous issues faced in federated learning can be resolved; the most notable improvement of which can be found in terms of scalability and can be decomposed into three distinct aspects:

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

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

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

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

Nodes in decentralized learning may not have access to all information in the network, causing a decrease in informativeness. More specifically, convergence speeds may decrease compared to federated learning [36] and nodes may experience increased vulnerability to byzantine attacks due to the lack of global information [24]. However, it has been shown that decentralized learning can obtain comparable performance results relative to federated learning and, in some cases, outperform federated learning altogether [37], [38].

The network graph $\mathcal{G}$ in decentralized learning, in contrast to that of federated learning, does not contain a parameter server, and nodes are generally not limited to only be connected through edges with a subset of the set of nodes N.

C. Targeted poisoning attacks

Poisoning attacks can be defined as methods with which an adversary attempts to compromise the integrity of the global model in a some form of distributed machine learning, and can be taxonomized into two categories: targeted poisoning attacks and untargeted poisoning attacks. With untargeted poisoning attacks, the adversary aims to decrease the performance metric of the model without any particular goal in mind. On the other hand, in the context of targeted poisoning attacks, the adversary aims to achieve a predetermined goal by manipulating the global model to behave in a certain deterministic manner which deviates from the objectively correct behaviour. We consider two of such attacks related to the domain of classification: the label-flipping attack [26], [27] and the backdoor attack [28]–[30].

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

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

The backdoor attack requires a more 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 small

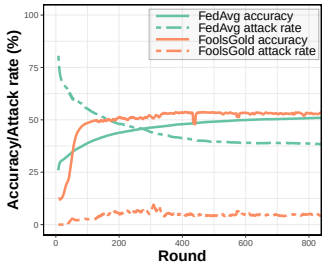
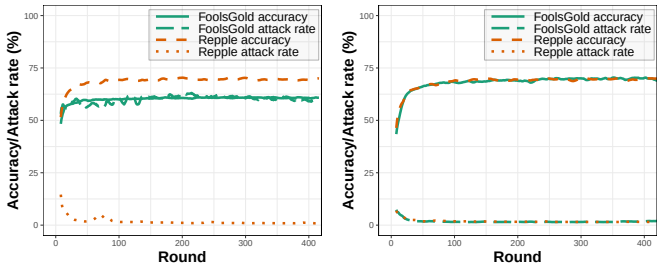


Figure 3: FoolsGold and FedAvg in federated learning setting using the CIFAR-10 [39] dataset on a LeNet-5 [40] model.



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

Figure 4: FoolsGold and SybilWall in decentralized learning using the FashionMNIST [41] dataset on a single-layer softmax neural network.

square or triangle in the top-left corner of the input image. Given a target class t and a function f to introduce a hidden pattern to input samples, the dataset transformation applied on the adversary's local dataset $\mathcal{D}$ can be defined as:

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

D. The Sybil attack

The Sybil attack, first introduced by Douceur [31], is an adversarial strategy in decentralized environments in which the attacker exploits the inability of honest nodes to verify the authenticity of another node's identity. By effortlessly creating fake nodes, named *Sybils*, and strategically connecting these to nodes in the targeted decentralized network, the attacker may gain significantly more influence compared to the honest nodes. We denote the connections between Sybils and honest nodes as *attack edges*. A typical example of a scenario in which the Sybil attack may be deployed is *voting* [42], [43]. In such a case, an attacker can easily generate sufficient nodes to outnumber all other real voters.

A network 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 a regular point of view. The modified set of edges is defined as $E' = E \cup E_{\mathcal{S}}$, where $E_{\mathcal{S}}$ is the set of *attack edges*, which is highly dependent on the strategy of the adversary. Note that attack edges always consist out of at least one Sybil, such that $\forall \langle i, j \rangle \in E_{\mathcal{S}}, i \in E_{\mathcal{S}} \vee j \in E_{\mathcal{S}}$.

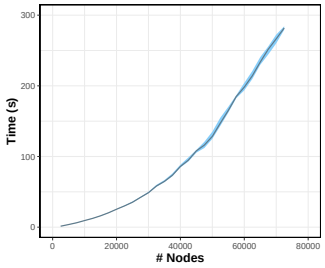


Figure 5: Pairwise cosine similarity computation time against the number of nodes, illustrating the $\mathcal{O}(n^2)$ time complexity of the pairwise cosine similarity computation. TODO: this graph demonstrates two limitations: memory and time

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

III. Related work

A. FoolsGold

FoolsGold [1] is an algorithm for mitigating Sybil poisoning attacks in federated learning settings. It builds on the assumption that Sybil model updates show a substantially higher degree of similarity relative to that of honest model updates. Through the computation of similarity between a node's model update history and that of others, and subsequently mapping this to the model update's weight in an average-based aggregation, FoolsGold manages to mitigate Sybil targeted poisoning attacks as shown in Figure 3.

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

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

A reproduction of FoolsGold's results can be found in Figure 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 the FedAvg. However, as discussed in Section II-B, federated learning can be considered unscalable as the

number of participating nodes increases, particularly in view of the $\mathcal{O}(n^2)$ pairwise cosine similarity computation required by FoolsGold. Figure 4 shows the performance of FoolsGold in a decentralized setting against the performance of our improved algorithm, SybilWall, based upon FoolsGold. It is clear that FoolsGold's performance can heavily depend based on the network topology, whereas SybilWall demonstrates relatively higher and more consistent Sybil resilience.

B. Krum

Krum [44] attempts to improve the general Byzantine resilience in distributed machine learning. This approach operates on the premise that Byzantine model updates are prone to deviate from the updates produced by honest participants. More specifically, the aggregation involves computing a score $s(i)$ for every received model i, which corresponds to the sum of the squared distance between i and its $n - f - 2$ nearest distant-wise neighbours, where f corresponds to the maximum number of Byzantine participants. The model m with the lowest score, such that $m = \arg\min_i s(i)$, is chosen as the next model to train on.

C. Resilient Averaging Gradient Descent

Resilient Averaging Gradient Descent (RAGD) [32] utilizes similar distance-based intuitions as Krum, but is specifically designed for decentralized learning. By introducing additional assumptions, it guarantees convergence of an approximately optimal model in the presence poisoning attacks in decentralized learning. Firstly, it assumes that all nodes are honest and that solely their local datasets might be compromised, thereby still participating in aggregation benevolently, but producing malicious models. Secondly, RAGD assumes the existence of a weighted global adjacency matrix in which the weights are considered "trust scores" and correspond to the influence nodes have during aggregation. RAGD assumes that the edge weights from some node i to some attacked neighbouring node j is limited by a predefined global constant ϵ, such that $0 < \epsilon < \frac{1}{2},\ a_{ij} < \epsilon$, where a_{ij} corresponds to the edge weight assigned by node i to its edge with attacked node j.

A typical round of training in RAGD can be decomposed in a number of steps. 1) Nodes attempt to reach a global consensus on the aggregated model through repeatedly broadcasting and averaging models aggregated by neighbouring nodes, weighted by the corresponding edge weight. 2) Every node trains the aggregated model and broadcasts the gradients. Note that malicious models may be produced by attacked nodes during this step. 3) While some value g, which is initialized to 1, remains larger than $1 - \epsilon$, RAGD selects two of the received gradients, such that the euclidean distance between the two selected gradients g_i and g_j is maximized, and eliminates the gradient which has the largest summed distance to all other gradients. The edge weight corresponding to the node which produced the eliminated gradient is subtracted from the value g. When $g \leq 1 - \epsilon$, every node computes the weighted average of the remaining gradients. 4) Finally, the weighted average of the remaining gradients is applied to the (pre-training) aggregated model and the next round begins.

IV. Threat model and preliminaries

A. Adversarial assumptions

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

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

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

We assume that the model updates by Sybils exhibit a higher degree of similarity compared to updates made by honest participants, as stated by prior work [1].

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

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

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

B. Network restrictions

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

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

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

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

C. Adversarial attack strategy

We define an intuitive and effective type of attack in similarity-based aggregation techniques in Decentralized Learning as the *Spread Sybil Poisoning Attacks* (SSP attack). That is, the adversary aims to evade detection by maximizing the distance between its attack edges while increasing the 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* [45], which is known to be an NP-Hard problem [46]. To determine attack edge positions for SSP attacks, we propose a heuristic approach using the K-medoids unsupervised clustering algorithm, assigning attack edges to the medoids.

Furthermore, we define a parameter for SSP attacks, ϕ, which represents the average density of attack edges per node. Note that in this distribution, 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. As such, if $\phi = 1$, we say that every honest node has exactly one attack edge. For any value of ϕ, each honest node receives either $\lfloor \phi \rfloor$ or $\lceil \phi \rceil$ attack edges. The total number of attack edges is therefore denoted as $\lceil |N| \cdot \phi \rceil$, where $|N|$ is the total number of honest nodes. The remainder, denoted as $\phi \bmod 1$, is distributed according to the K-medoids clustering algorithm. The resulting attack edge positions are then grouped and distributed over the Sybils while upholding Assumption 5. We define three attack scenarios for specific ranges of ϕ. These attack scenarios are the following:

i. Dense Sybil poisoning attacks. $\phi \geq 2$. Every honest node has at least two attack edges, whereas any distinct Sybil node cannot form more than one attack edge to any given node. As a result, the honest node is a direct neighbour of at least two distinct Sybil nodes.
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 node.
iii. Sparse Sybil poisoning attack. $\phi \leq \epsilon$. A low value of ϕ will result in sparse and distant attack edges. Nodes which are a direct neighbour of a Sybil node are rare. Any node has a probability of ϕ being directly connected to a Sybil node.

V. Design of SybilWall

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

In short, SybilWall was designed to mitigate the three attack scenarios of the worst-case attack scenario, as listed in Section IV-C. These scenarios are the following:

i. Dense Sybil poisoning attacks: The adopted FoolsGold subfunction can detect Sybil nodes directly through its similarity mechanism, thereby capable of mitigating the attack.
ii. Distributed Sybil poisoning attack: Not all nodes are capable of detecting potential Sybils among its direct neighbours through a similarity metric. The Sybil updates it receives will likely vary from all other received updates and therefore considered honest by Assumption 2. Our probabilistic gossiping mechanism serves as a channel for data dissemination by propagating data of probabilistically selected nodes to neighbours, thereby potentially providing a neighbour with vital data required for detecting Sybils amongst its neighbours.
iii. Sparse Sybil poisoning attack: Due to the low density of Sybils, it is conceivable that the probabilistic gossiping mechanism may be unable to disseminate knowledge to an extent that enables all attacked nodes to detect the presence of Sybils prior to the completion of the training process. Another possible scenario in which attacked nodes may not obtain sufficient knowledge to successfully detect Sybils among their neighbours is due to the natural divergence of the data produced by Sybils. In such case, the information received by the attacked nodes should theoretically enable them to detect Sybils, but may be outdated to such an extent that the information has sufficiently diverged for both Sybils to be considered honest by Assumption 2. In both aforementioned scenarios, we argue that due to a natural dampening effect originating from the train-aggregate loop on each node, depicted in Figure 2, an attack's effect will likely result in a negligible and tolerable effect on the average global model.

A. Adopting FoolsGold

Given Assumption 2 (Sybil models are likely to show high similarity) and the promising performance of FoolsGold [1] on exploiting this increased similarity in a federated learning setting, we include a modified version of FoolsGold as a subfunction within SybilWall. This subfunction's main task is to mitigate dense Sybil poisoning attacks, as it will know of the existence of at least two attack edges and will thereby exclude them from the aggregation due to their high similarity. However, this subfunction also plays a role in mitigating distributed Sybil poisoning attack, with the help of the probabilistic gossiping mechanism. We assume that the reader is aware of the internal workings of FoolsGold, as described in Section III-A.

Firstly, we modify FoolsGold by always trusting the aggregator. There is no need to compare the similarity with the current aggregator, as we assume that a node's training dataset and their training function cannot be compromised, and can therefore trust itself. As such, we exclude the aggregator's model from the cosine similarity and logit scoring function and introduce it prior to the normalization of the weights with a weight of 1. An additional argument for this design choice

is that neighbours with similar datasets, and therefore similar models, should not be penalized during aggregation.

Secondly, FoolsGold is adapted to support the addition of an arbitrary amount of model histories on the cosine similarity function. The purpose of the gossiping mechanism described in Section V-B is to spread information about indirect neighbours by communicating their model histories. By including these in the cosine similarity function of the FoolsGold subfunction, SybilWall potentially gains the ability to detect new Sybils among its direct neighbours, thereby mitigating their attacks from that point onwards. Note that only the models of direct neighbours are considered for aggregation and are merely judged based on the additional information obtained through gossiping.

B. Probabilistic gossiping mechanism

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

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

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

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

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

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

A node's local database of model update histories can be updated in a number of manners. Firstly, through direct neighbour updates. During every round of training, every node receives a model update from their direct neighbours and update their model update history of that neighbour accordingly. Secondly, if a node receives a the history of model updates through gossiping which it has not seen before, it is registered directly. Lastly, if a node j receives a history of model updates from some node i which is more recent than the prior history of model updates known to node j, it is updated accordingly. Note that a node's local collection of model update histories may grow to a significant size over time, resulting in a decrease in performance during aggregation. In such scenarios, our gossiping mechanism supports dropping outdated model update histories to mitigate this occurrence.

Mention synchronous training rounds?

2) Secure and efficient communication: SybilWall replaces the traditional model sharing discussed in Section II-B with a more sophisticated communication protocol. The previously described probabilistic gossiping mechanism requires model histories to be propagated to neighbours, which allows for spoofing by malicious nodes if implemented naively. Such strategies could be employed to increase some target node's similarity with another node, and thereby increasing the probability for a higher attack score by the adversary. To mitigate this vulnerability, we propose an alternative communication protocol employing cryptographically secure signatures.

To enable the use of signatures, model update histories need to be constructed on the originating nodes and communicated to its neighbours, which can then propagate it to their neighbours according to the probabilistic gossiping mechanism. However, this induces additional communication costs on any node as both the model and the model history need to be communicated to its neighbours. We resolve these redundant communication costs by only sending the the model history. The trained model itself can be inferred from the model history by comparing it with the previous model history in the sequence.

todo: define time t for history More specifically, we construct messages such that a message $m_{i \to j}$ from node i to j can be decomposed into $\langle h_i, S_i(h_i), g_k, S_k(g_k) \rangle$, where h_i represents node i's updated model history signed by its signature function S_i and g_k corresponds to node k's gossiped model history signed by node k.

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

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

Dataset	Model	Learning rate
MNIST [47]	Single soft-max layer	$\eta = 0.01$ [1]
FashionMNIST [41]	Single soft-max layer	$\eta = 0.01$ [1]
CIFAR-10 [39]	LeNet-5 [40]	$\eta = 0.004$ [48]
SVHN [49]	LeNet-5 [40]	$\eta = 0.004$ [48]

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

Hyperparameter	Value
# honest nodes	99
Attack edge density ϕ	1
Gossip mechanism parameter λ	0.8
Dirichlet concentration parameter α	0.1
Max node degree d	8
Local epochs	10
Batch size	8

One may note an alternative for the altered communication protocol described previously in which we use a pull-based history. However, the disadvantage of this approach is the reachability of these nodes.

VI. Evaluation

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

A. Experimental setup

SybilWall was implemented in Python3 for experimental evaluation and is online available [50]. We have used the PyTorch [51] library for the training of machine learning models. As for the communication between the individual nodes, we leveraged IPv8 [52], which provides an API for constructing network overlays in order to simulate P2P networks. Furthermore, we adopted the Gumby [53] library as the experimental execution framework, which was specifically designed for sophisticated experiments with IPv8 involving many nodes. All experiments were performed on the Distributed ASCI Supercomputer 6 (DAS-6) [54]. Each node in the compute cluster has access to a dual 16-core CPU, 128 GB RAM and either an A4000 or A5000 GPU. Furthermore, all default hyperparameters for the experiments can be found in Table II. Except where mentioned otherwise, these default hyperparameters define the configuration of all experiments.

In all experiments, we measure the accuracy of the trained models by averaging the accuracies of the models of all honest nodes. Simultaneously, we measure the success rate of the attacker by averaging the attack score achieved on all models. The attack score is defined as the accuracy a model obtains on the test dataset transformed by the data transformation defined in either equation 4 or 5. Note that both metrics are measured each round directly after aggregation.

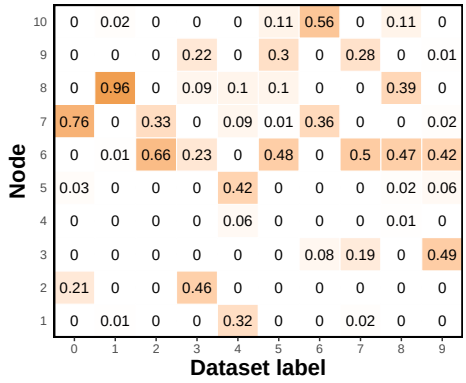
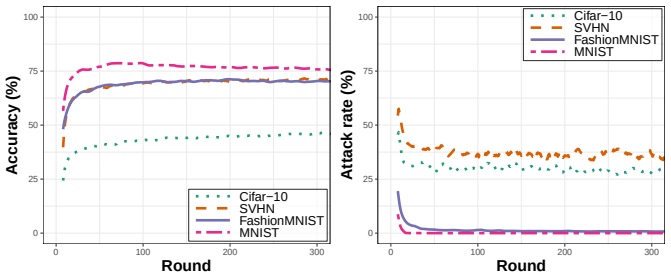


Figure 6: Example distribution for non-i.i.d. data generated with the Dirichlet distribution with concentration parameter $\alpha = 0.1$ for 10 nodes and a dataset containing 10 labels, such as CIFAR-10 [39] or MNIST [47].

1) Datasets: The datasets used during evaluation can be found in Table I. These datasets were chosen for a number of reasons. First of all, MNIST [47] is a broadly-used dataset for the evaluation of machine learning algorithms [1], [55]–[57], serving as an adequate baseline algorithm for SybilWall. FashionMNIST was developed as a more challenging variant of MNIST, thereby providing an ideal candidate for demonstrating 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 models required to obtain satisfactory accuracy, which may affect the performance of SybilWall. The usage of complex multi-layer models in evaluation is frequently overlooked in related work or only perform an evaluation on a single dataset [1], [55]–[59]. Moreover, in the case when multi-layer models are used, they are regularly pre-trained and solely evaluated through the use of transfer learning [1], [57], [60]. While recognizing that all the datasets employed in this experimental evaluation focus on image classification, we argue that that focusing on image classification is justifiable as it is known as a well-established task in machine learning. Furthermore, image classification frequently serves as a benchmark for evaluating novel distributed machine learning algorithms [1], [36], [55]–[59], and there exist a variety of widely available datasets constructed specifically for this task.

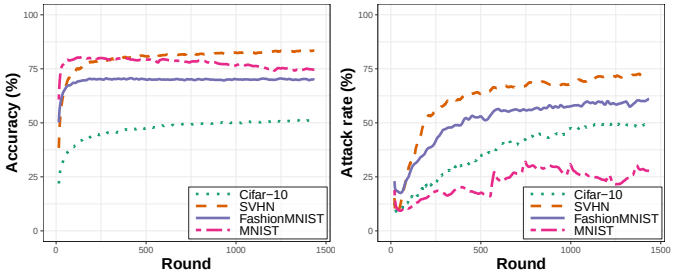
The models which are trained using the aforementioned datasets can also be found in Table I, as well as the corresponding learning rate η. Note that all these models are trained using stochastic gradient descent (SGD).

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 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) [61], [62]. While some works employ the use of a K-shard data distribution [9], [57], [63], [64] or simply assign each node a



(a) Accuracy label-flipping attack

(b) Attack score label-flipping attack



(c) Accuracy backdoor attack

(d) Attack score backdoor attack

Figure 7: Accuracy and attack score for the label-flipping attack (300 rounds) and the backdoor attack (1450 rounds) on different datasets.

defined number of classes of the training data [1], [63], [65], we utilize the Dirichlet distribution [66], which has recently gained more popularity for generating non-i.i.d. distributions [36], [67], [68]. More specifically, given the *concentration parameter* α, we compute for each class the fraction of data every node possesses, creating seemingly naturally unfair and irregular data distributions. Figure 6 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 honest nodes, we employed *random geometric graphs*. Random geometric graphs are are constructed by randomly placing points, which correspond to nodes, in a grid. Two nodes are connected by an edge if and only if the euclidean distance between the corresponding points of these nodes is smaller than some predefined constant. To enforce the max node degree bound d, random edges are removed from the random geometric graph, such that all honest nodes remain connected through a single connected component. The locations of attack edges are found using the K-medoids-based methodology described in Section IV-C. The code used for generating these network topologies can be found in our published code repository [50]. Furthermore, during our experiments, we assume a static network topology. That is, no nodes will leave or join the network during training, including any Sybils.

B. Effect of dataset

1) Setup: We evaluate SybilWall's performance on different datasets, allowing us to observe how SybilWall is affected by a varying complexity in both the dataset and the model. Note that this experiment has been performed using the default parameters listed in Table II and using the datasets, models and learning rates listed in Table I.

2) Results: Figure 7 demonstrates the effect of varying the dataset on the converged accuracy and attack score. We observe clearly that CIFAR-10, arguably the most challenging dataset in this work, obtains a significantly lower accuracy compared to simpler datasets, such as MNIST. This can be explained by the fact that training samples of easier datasets have less overlap over the different output classes, and individual nodes are therefore less likely to counteract their neighbours.

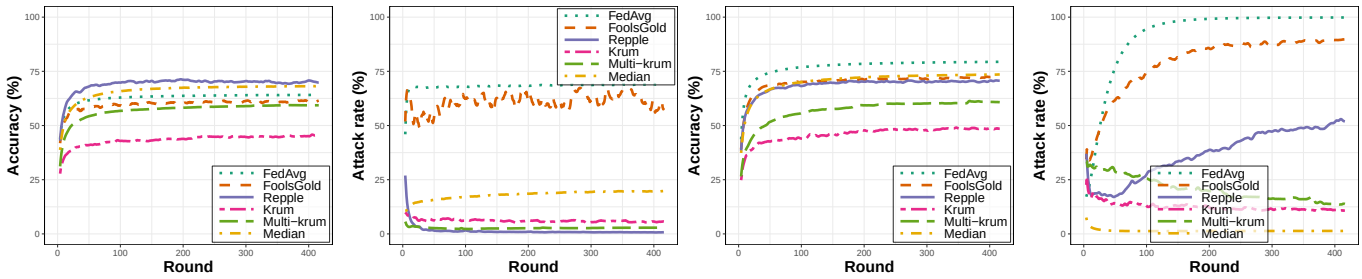
A noteworthy observation with regard to the attack score of the label-flipping attack in Figure 7a is that the datasets requiring more sophisticated models, such as convolutional neural networks, are generally more susceptible to the label-flipping attack compared to the simpler models, such as single layer neural networks. Due to smaller number of trainable weights in simpler models, it is easier to distinguish similarities between Sybil model histories of simpler models compared to more sophisticated models, which show more diversity. While some fraction of the higher attack score in complex models can be attributed to the more challenging classification task, later experiments show that the attack score can be significantly reduced under certain conditions (Sections VI-E and VI-F).

Considering the results from the backdoor attack depicted in Figure 7c and 7d, it is apparent that all attack scores demonstrate an increasing trend over a longer amount of time (note the difference between the number of rounds between the evaluation on the label-flipping attack and the backdoor attack attack). This finding suggests that the aggregation technique performed to obtain a node's model history does not always provide a reliable reflection of the node's intentions. However, the time required for the attack score to achieve convergence is significantly longer than the time required for convergence of the accuracy for most datasets. Therefore, it can be argued that nodes should stop the training process once the accuracy has converged or decreases.

C. Comparison with different techniques

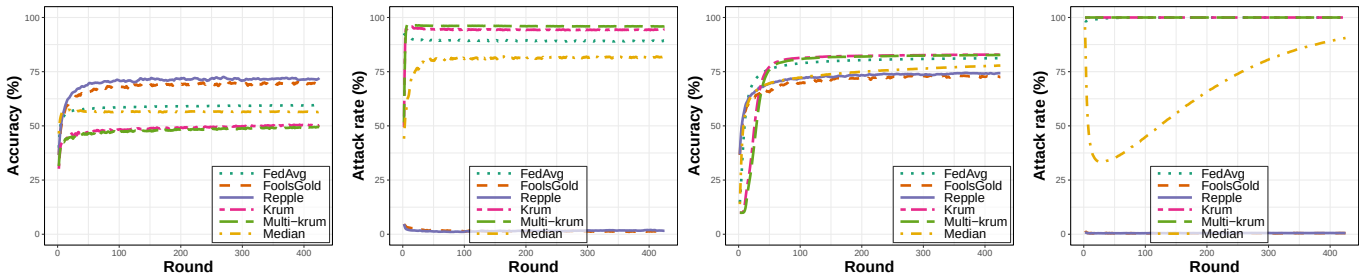
1) Setup: We evaluate SybilWall's performance relative to a number of different techniques focused on mitigating Sybil poisoning attacks or Byzantine attacks in general. These techniques are the following:

i. FedAvg [9]: Naively averages all models. This algorithm was the first proposed federated learning aggregation algorithm and will serve as baseline during our experiments.
ii. FoolsGold [1]: Detects Sybil models among its neighbours by assuming that Sybil model histories demonstrate high similarity. This algorithm is the main inspiration for SybilWall.
iii. Krum [44]: A popular algorithm which aims to exclude byzantine models by selecting the model which has the



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

Figure 8: Comparison of SybilWall against different techniques on $\phi = 1$. Results generated using the FashionMNIST [41] dataset.



(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 $\phi = 4$. Results generated using the FashionMNIST [41] dataset.

smallest sum of euclidean distances to its $n-f-2$ closest neighbours.

iv. Multi-krum [44]: Similar to krum, but averages the m models with the lowest sum of euclidian distances to its $n-f-2$ closest neighbours.

v. Median [69]: Computes the element-wise median of all models and thereby excluding outliers.

During this experiment, we alternated the attack edge density between $\phi = 1$ and $\phi = 4$, and fixated the dataset to FashionMNIST.

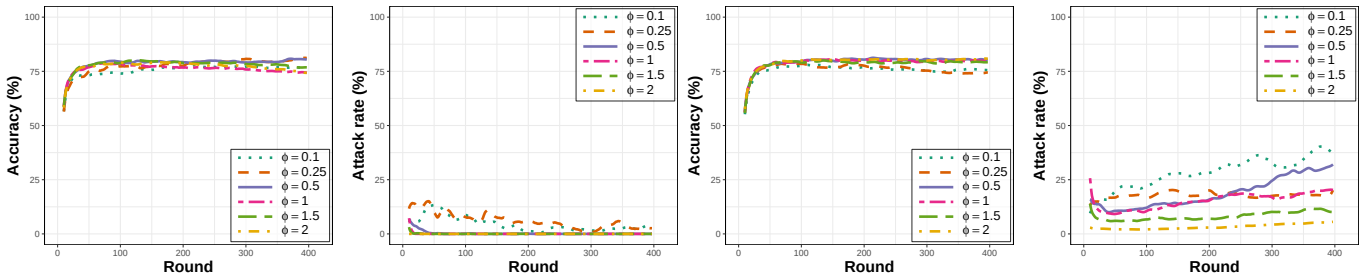
2) Results: Figure 8 shows the results of SybilWall compared to different techniques using an attack edge density $\phi = 1$. We observe that SybilWall always scores among the best performing algorithms in terms of accuracy. Especially considering the label-flipping attack, SybilWall achieves the highest accuracy among all evaluated techniques. Considering the label-flipping attack, we find that SybilWall successfully mitigates the attack, similarly to some of the other evaluated techniques. In the backdoor attack, we observe the same increasing pattern as in the previous experiment on the effect of the datasets in Section VI-B; the attack score starts at a low point and increases gradually as the training progresses. However, while the *median* aggregation algorithm undoubtedly achieves the strongest backdoor attack mitigation, we claim that SybilWall still performs better than other algorithms, as the accuracy has converged by the time the attack score reaches significantly dangerous values.

Figure 9 shows the results of SybilWall compared to different techniques using a higher attack edge density of $\phi = 4$. These results clearly demonstrate how most aggregation algorithms succumb under the employment of a large-scale Sybil attack. Considering the accuracy of both the label-flipping attack and backdoor attack, we observe that the accuracy of most algorithms is higher when employing the backdoor attack. This phenomenon can be explained due to the fact that the adversary is not actively attempting to decrease the accuracy of the model, but only tries to insert an activation pattern, which was highly successful these algorithms. However, both FoolsGold and SybilWall seem to be wholly unaffected by both attacks. This is due to the FoolsGold subfunction, which was specifically designed to mitigate Sybil poisoning attacks on nodes with knowledge of more than a single Sybil, which is evidently the case for an attack edge density of $\phi = 4$.

Considering both results in Figure 8 and 9, we find that SybilWall does not outperform all evaluated algorithms in every scenarios, but always scores among the best, arguably making it the most Sybil poisoning resilient algorithm overall. A surprising observation is that most other algorithms score better under a backdoor attack with a high attack edge density compared to a lower attack edge density, while the accuracy of SybilWall remains constant in both scenarios.

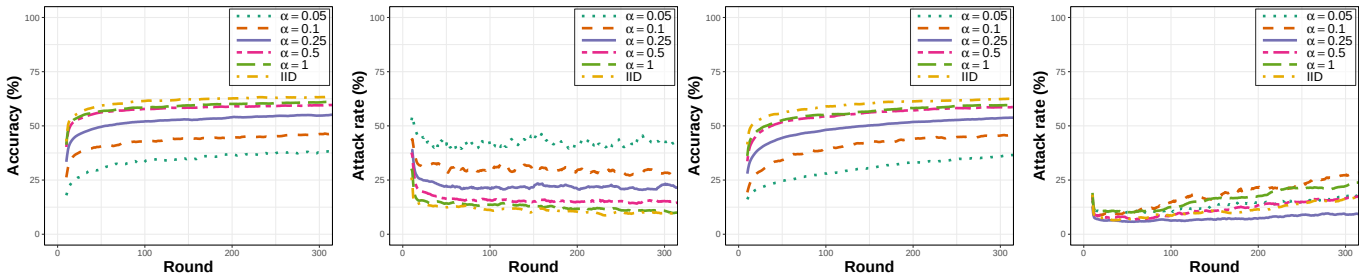
D. Effect of attack edge density

1) Setup: We evaluate SybilWall under a number of different attack edge density configurations. This demonstrates



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

Figure 10: Accuracy and attack score for the label-flipping attack and backdoor attack on different attack edge densities. Results generated using the MNIST [47] dataset.



(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 of different data distributions, indicated by the concentration parameter α of the Dirichlet distribution. Results generated using the CIFAR-10 [39] dataset.

the effect an attacker can exercise on the network employing different Sybil attack strategies. MNIST is fixated as the dataset during this experiment. The attack edge density ϕ is varied within the range $\phi \in [0.1, 2]$.

2) Results: Figure 10 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. On the other hand, the trend of the attack score shows that network topologies with lower attack edge densities are more prone to the label-flipping attack despite the smaller number of generated Sybils for lower values of ϕ. This clearly demonstrates the effect of the gossiping mechanism in reducing the impact of Sybil poisoning attacks.

Similarly to the label-flipping attack, we observe the success of SybilWall's gossiping mechanism as the attack score generally decreases as the attack edge density grows. Furthermore, as one would expect, the accuracy also does not seem to be affected by different values of ϕ.

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 α within the range $\alpha \in [0.05, 1]$. Furthermore, we fixate the dataset on CIFAR-10.

2) Results: Figure 11 shows the effects of different concentration parameters α on the convergence of the training process on the CIFAR-10 dataset. We observe in both the label-flipping attack and backdoor attack that the accuracy increases as the data is more uniformly distributed, as one would anticipate. Furthermore, the attack score of the label-flipping attack demonstrates how the attacker becomes less successful with more i.i.d. data. On the one hand, one would expect a weaker Sybil resilience when the data is less non-i.i.d. resulting from false positives among honest nodes. However, we found that nodes more easily counteract Sybils when they possess data belonging to an adversary's target classes. Lastly, the data distribution surprisingly does not seem to have a significant effect on the attack score of the backdoor attack, as no clear trend emerges when varying the data distribution.

F. Enhancing SybilWall's defensive capabilities

1) Setup: Given the increasing, although impeded, attack score demonstrated for the backdoor attack in Section VI-B for more complex datasets, we consider several augmentations for improving SybilWall's defensive capabilities. These augmentations include:

i. Median: given the resilience of the Median [69] algorithm in Section VI-C against attack edge density $\phi = 1$, we attempt to combine the Median approach with SybilWall. This is achieved by initially employing SybilWall to

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

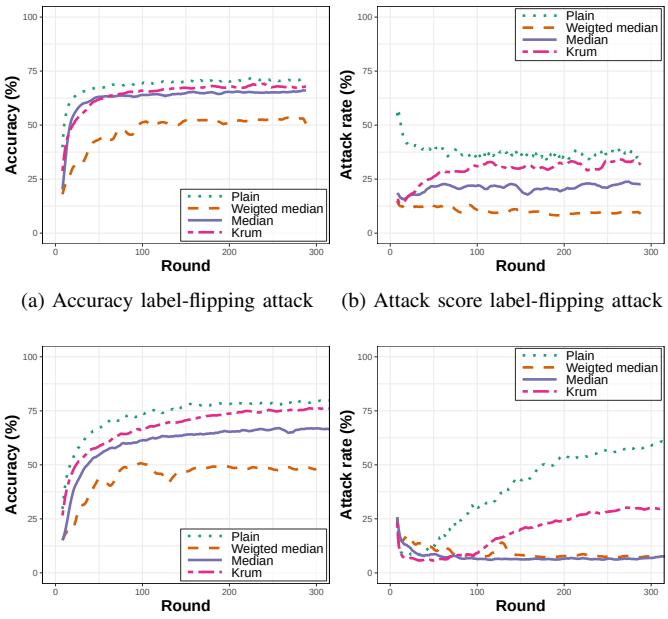
ii. Weighted median: A variant of the Median-based approach, in which scores computed by SybilWall are adopted as weights for a weighted median aggregation.

iii. Krum-filter: based on the suggestion by [1], we combine SybilWall with Krum, such that the model with the lowest Krum score receives an aggregation weight of 0.

iv. Distance-filter??? (if we have time). Only consider the 50% closest models (inspired by [57]?).

We also provide the trends for plain SybilWall to serve as a baseline. The dataset is fixated to SVHN during this experiment.

2) Results: Figure 12 illustrates the effect of enhancing SybilWall with a number of methodologies. Firstly, 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 improved Sybil resilience. Especially the Sybil resilience of the weighted median is unmatched, but achieves considerably lower accuracy compared to the alternative methodologies. The Krum-filter based approach seems to obtain comparable accuracy as plain SybilWall, but obtains the worst Sybil resilience of the evaluated enhancements. Arguably, the median-based methodology shows the most promising results, as it achieves to consistently limit the attack score to levels comparable of that of the weighted median methodology, while showing significant improvement on the obtained accuracy.



(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 of the label-flipping attack and backdoor attack for different possible enhancements of SybilWall. Results generated using the SVHN [49] dataset.

VII. Discussion

During the experimental evaluation of SybilWall, we found that SybilWall does not negatively affect the convergence of the accuracy on 4 widely adopted datasets. Moreover, the converged accuracy obtained by SybilWall is similar to that achieved by the FedAvg algorithm in a federated learning setting (see Figures 3 and 7). Besides obtaining satisfactory accuracy on all datasets, we also found that SybilWall significantly outperforms alternative algorithms for mitigating (Sybil) poisoning attacks in decentralized learning in both obtained accuracy and attack score. While the attack score of the backdoor attack shows a rising trend, we assert that the rate at which this occurs is sufficiently diminished to attribute SybilWall with the required performance to be regarded as state-of-the-art.

We argue that this rising trend originates from the knowledge loss induced by the train-aggregate loop depicted in Figure 2. Building upon Assumption 2 (Sybils show similar models), the implicit assumption is made that summed Sybil model histories show a high degree of similarity as well. It would arguably be more effective to sum the model post-training gradients to generate a node's history, rather than the model itself. Such an approach would provide a more accurate representation of a node's intentions as the model history would directly correspond with a node's training data and would show in which *direction* a node aims to contribute to the aggregated model. As this representation does not include the aggregated model, it is also less affected by neighbouring nodes. However, obtaining a node's post-training gradients is a non-trivial task in the setting of decentralized learning, as there exists no method of validating the aggregated model which was trained to generate the gradients without sharing the corresponding training data. As an example, a malicious node could claim to start training on a randomly generated model m_r which resulted in the training gradients g, such that the sum of these is equal to the malicious model $m_s = m_r + g$. An adversary could trivially manipulate the sums of gradients to make Sybil models seem highly diverse. This drawback is absent in federated learning, as the model trained by nodes is equal for all nodes every round and is not created through aggregation on participating nodes (see Figure 2). Through making additional assumptions, a methodology for alleviating this drawback is described in Section VII-A.

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

During the evaluation of the effect of the number of Sybils on the attack score in Section VI-D, we found that decreasing the number of Sybils increases the attack score. This implies that we eliminated the need of amplifying a poisoning attack with the Sybil attack, as employing Sybils would result in a lower attack score. Reducing the Sybil poisoning attack to a mere poisoning attack, which cannot be deflected by SybilWall, requires the usage of alternative poisoning attack

mitigation algorithms. During evaluation, we considered enhancing SybilWall with a number of such alternative algorithms through the use of chained aggregation methodologies in Section VI-F. In such a chained aggregation, the last step of SybilWall's aggregation method, a weighted average, is substituted with one of the listed methodologies. We found that while all enhancements demonstrate an increased Sybil resilience, they all require a sacrifice in accuracy, likely due to the increased number of false positives among individual weights caused by the non-i.i.d training sets. Considering that accuracy is often the primary goal in machine learning [70], the usage of such enhancements is likely not justified in most applications. We leave enhancing SybilWall with a poisoning attack mitigation algorithm without compromising accuracy as a possible research direction for future work.

Furthermore, SybilWall has been designed under Assumption 2, which states that Sybil model histories show more similarity compared to honest model updates. We found that SybilWall shows more Sybil resilience in more i.i.d. environments in Section VI-E, despite the fact that all honest nodes will share more similar models, thus potentially resulting in more false positives. However, it is unclear if SybilWall shows sufficient resilience against adversaries violating Assumption 4, stating that the creation of additional Sybils by the adversary does not increase the adversarial computing power. Adversaries with extensive computational capabilities, such as click-farms [71], may well be able to train several malicious models within one training round, thereby possibly violating Assumption 2. A possible future research direction for alleviating this risk is proposed in Section VII-A. Another method of creating more diverse Sybil model histories is through the introduction of random noise in the irrelevant weights of the model, as suggested by [1]. More research is required to filter out these irrelevant weights, which may be achieved through a number of approaches, such as layer-wise relevance propagation [72], weight magnitude filtering [73] or empirical weight importance [74].

A. Future research: replicated aggregation

As described previously, SybilWall's Sybil resilience against backdoor attack could potentially be drastically increased through the use of summed model gradients as a model's history instead of summed models, as it provides a more direct representation of a node's intentions. To obtain the model gradients, one requires the ability to obtain the intermediary model between aggregation and training. However, it is challenging to obtain this pre-training model through reverse-engineering, especially considering that no node has access to another node's training dataset. Instead, we suggest an alternative direction, which entails replicated aggregation. Using this approach, both the neighbours of node i and node i itself perform the aggregation of node i. By doing so, neighbouring nodes gain the power to determine the intermediary model between aggregation and training.

First, this adapted approach assumes the existence of a global registration, containing cryptographically secure public records of all edges in the network, such as a blockchain ledger, e.g. TrustChain [75]. This allows any node to obtain the identities of their second neighbours, without adversaries being able to provide false or inconsistent information about its neighbours. Furthermore, all nodes must possess all required information to perform aggregation for its neighbours' models, implying that any node requires direct communication with its second neighbours. Once node i possesses all information required to perform its own aggregation, it starts training the aggregated model.

After the training has completed, every node i sends the gradients and its aggregated model to its set of neighbours n_i and second neighbours n_i'. Consequently, every neighbour $j \in n_i$ sends their own version of the aggregated model of node i to its neighbours $n_j \setminus i \ (= n_j \cap n_i')$. This allows node $k \in n_i'$ to verify the aggregated model on j's behalf and obtain the trained model of node i required for the aggregation for their shared neighbour(s) $n_k \cap n_i$ without requiring them to perform the aggregation of node i as well. By doing so, node j gains no advantage in providing node k with a malicious model over simply performing malicious aggregation in *plain* decentralized learning, as it only affects its own aggregation.

While this approach suggests a method of obtaining the aggregated model and thereby enables (a variant of) SybilWall to compose the model history by the sum of trained gradients, future work is required to evaluate the effectiveness and identify potential drawbacks of this method, other than the increased communication load.

VIII. Conclusion

References

[1] 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

[2] 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.

[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] 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.

[6] 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.

[7] 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

[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] 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

[24] 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.

[25] 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.

[26] 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

[27] 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.

[28] 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

[29] 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

[30] 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

[31] 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.

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

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

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

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

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

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

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

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

[40] 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.

[41] 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

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

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

[44] 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.

[45] 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.

[46] 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

[47] 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.

[48] 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

[49] 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

[50] T. Werthenbach, "Sybil-resilient-decentralized-learning," https://github.com/ThomasWerthenbach/Sybil-Resilient-Decentralized-Learning, 2023.

[51] 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

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

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

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

[55] 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.

[56] 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

[57] 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

[58] 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.

[59] 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

[60] 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.

[61] 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.

[62] 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

[63] 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

[64] 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.

[65] 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.

[66] 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

[67] 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.

[68] 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

[69] 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

[70] 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.

[71] 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.

[72] 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

[73] 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.

[74] 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.

[75] P. Otte, M. de Vos, and J. Pouwelse, "Trustchain: A sybil-resistant scalable blockchain," *Future Generation Computer Systems*, vol. 107, pp. 770–780, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X17318988