On the (In)security of Peer-to-Peer Decentralized Machine Learning

Dario Pasquini
SPRING Lab; EPFL, Switzerland
dario.pasquini@epfl.ch

Mathilde Raynal
SPRING Lab; EPFL, Switzerland
mathilde.raynal@epfl.ch

Carmela Troncoso
SPRING Lab; EPFL, Switzerland
carmela.troncoso@epfl.ch

2023 IEEE Symposium on Security and Privacy (SP) | 978-1-6654-9336-9/23/$31.00 ©2023 IEEE | DOI: 10.1109/SP46215.2023.10179291

***Abstract*—In this work, we carry out the first, in-depth, privacy analysis of Decentralized Learning—a collaborative machine learning framework aimed at addressing the main limitations of federated learning. We introduce a suite of novel attacks for both passive and active decentralized adversaries. We demonstrate that, contrary to what is claimed by decentralized learning proposers, decentralized learning does not offer any security advantage over federated learning. Rather, it increases the attack surface enabling *any* user in the system to perform privacy attacks such as gradient inversion, and even gain full control over honest users' local model. We also show that, given the state of the art in protections, privacy-preserving configurations of decentralized learning require fully connected networks, losing any practical advantage over the federated setup and therefore completely defeating the objective of the decentralized approach.**

***Index Terms*—Collaborative Machine Learning, Privacy attacks, Peer-to-Peer systems**

1. Introduction

Collaborative machine learning is gaining traction as a way to train machine learning models while respecting the privacy of users' local training dataset [40]. There are two main approaches to collaborative machine learning: *federated learning* [40] and *decentralized learning* [36].

In *federated learning*, the iterative learning process is orchestrated by a central parameter server. This server intermediates communication in-between users and maintains the global state of the system. Such central component can become a communication bottleneck as the number of users grows, and, due to its full control on the learning process, can perform a number of security and privacy attacks on users [3], [15], [50], [64], [16], [79].

Decentralized machine learning, also known as fully-decentralized machine learning, peer-to-peer machine learning, or gossip learning, aims at addressing these issues by *eliminating the central server*. Instead, the learning takes place via peer-to-peer communication, see Figure 1. Proponents of decentralized learning argue that decentralization: *(a)* reduces bandwidth consumption, *(b)* provides users with control on who they communicate with, and *(c)* increases privacy of users in the system by eliminating the central server. A large body of theoretical studies, empirical evaluations, and model extensions attest to *(a)* and *(b)* [7], [23], [25], [30], [31], [32], [34], [35], [38], [49], [51], [52], [61], [36], [66], [72], [73], [62], [20]. However, these works do not assess *(c)*. Either they state that decentralized learning offers a higher level of privacy compared to the centralized approach without any evidence [7], [61], [20], [39], [58], or simply do not provide any privacy argument [25], [32], [34], [35], [36], [66], [72], [62], [11].

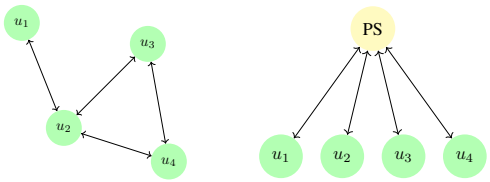
In this work, we thoroughly evaluate the privacy offered by decentralized learning, against both passive and active adversaries. We propose novel attacks that demonstrate that in a decentralized setting: **(1)** A passive adversarial user can successfully (i) *infer membership of samples* with better accuracy than in the federated setting and (ii) *perform reconstruction attacks* on the training set of arbitrary honest users. **(2)** An active adversarial user can (i) *influence the update process of honest users* in arbitrary ways and (ii) and perform effective privacy attacks such active gradient inversion [64], [3].

We show that these attacks are possible because decentralization *increases* the inference power of users, as well as their influence on other users' status. **This leads to adversarial users in decentralized learning becoming as powerful as the parameter server in federated learning.**

We study the effectiveness of mitigation techniques against our attacks. Our findings are two-fold. We first show that the potential protections against our attacks are in conflict: trying to eliminate one leakage factor augments another, leaving little space to eventually develop truly privacy-preserving decentralized learning. Second, we show that, while it is possible to reduce the attack surface resulting from decentralization, e.g., by changing the underlying topology and using expensive aggregation techniques, the privacy provided by decentralized learning will always be less (or equal at best) than the one provided by the federated counterpart. This invalidates claim *(c)*. Moreover, achieving protection comparable to federated learning comes at a huge cost in efficiency that destroys any remaining advantage of decentralization, invalidating claims *(a)* and *(b)*.

In summary, in contrast to common belief, in collaborative learning decentralization does not increase privacy. Instead, it inherently boosts the capabilities of privacy at-

DOI 10.1109/SP46215.2023.00175



(a) Decentralized Learning (b) Federated Learning

Figure 1: Schematic representation of the decentralized learning and federated learning protocols. "*PS*" stands for Parameter Server.

tackers and thus, decentralized learning tends to degrade users' privacy compared to the federated setting. This disadvantage cannot be overcome by existing mitigations without sacrificing the gains of decentralization over federated learning.

Our work makes the following contributions:

- We provide the first in-depth privacy evaluation of decentralized learning, in which users exchange updates in a peer-to-peer fashion. We characterize the key factors that contribute to privacy leakage within the protocol. We explore how these elements interact with each other to comprehend their effects on the privacy of decentralized users.
- We introduce a suite of novel privacy attacks designed specifically for the decentralized setting, covering both the passive and active security models. Our attacks demonstrate that decentralized users can reach the same adversarial capabilities of a parameter server in federated learning.
- We show that the capabilities that decentralization grants to adversaries results in a degradation of users' privacy compared to federated learning. Moreover, we show that no existing protection can bridge this privacy gap without eliminating the advantages of decentralization.

2. Preliminaries and Setup

In this section, we provide the necessary background on decentralize learning, we introduce the notation used in the paper, and we define our evaluation setup. We start in Section 2.1 by covering the Decentralized and Federated Learning protocols. In Section 2.2, we formalize the privacy attacks upon which we build our analysis and comparison.

2.1. Collaborative Machine Learning

We define Collaborative Machine Learning (CML) as the class of learning algorithms that enable a set V of n distributed *users* to train a shared model f defined by a set of parameters Θ. Each client $v \in V$ participates to the protocol with a local training dataset X_v. The local training sets must be kept private during the training. Thus, not only the training data cannot be shared directly, but the parties involved in the CML training must not be able to learn anything about each other's training set from their interactions, besides the information obtainable from the final model. Any additional information that an adversary can learn from observing or participating in the collaborative training constitutes a *privacy leakage*.

In CML, the model f is typically trained using a distributed version of Stochastic Gradient Descent (SGD), where users iteratively propagate *model updates*. These updates are intermediate outputs of each user's local optimization process such as gradients or updated parameters [40], [36], obtained after one (or more) local SGD step computed using their training set. A CML protocol defines the way in which model updates are shared among the parties.

2.1.1. Decentralized Learning. In Decentralized Learning (DL) [36], users connect to each other in a peer-to-peer fashion. During the protocol, every user $v \in V$ connects with a non-empty set of other users which we call *neighbors*, $\mathbf{N}(v)$, where $v \in \mathbf{N}(v)$. This set is typically small compared to the set of all users, and can either be fixed at the beginning of the execution or dynamically change across iterations. We model the communication links shared among users as an *undirected* graph $G = \{V, \cup_{v \in V} \mathbf{N}(v)\}$, where users are nodes and communication links are edges (see Figure 1a). Hereafter, we refer to this graph as *communication topology* or *topology* for short.

During the DL training, users share model updates only with their set of neighbors $\mathbf{N}(v)$. Through a gossip-like propagation mechanism, DL protocols ensure that users indirectly receive and benefit from the model updates produced by non-neighbor users.

In contrast to federated learning (see below) where connections are fixed and proxied by a server, DL protocols tend to not constrain the users' choice of neighbors. This flexibility is claimed as an advantage of DL as it enables users to cluster according to arbitrary criteria, e.g., data similarity or computational capabilities [1], [75]. In the general case, therefore, users participating in DL protocols can arbitrarily pick their neighbor nodes and autonomously define the underlying topology G. It is worth mentioning that a significant body of research has been dedicated to examining the effects of fixed and pre-defined topologies in decentralized learning. However, these studies primarily focus on theoretical aspects and do not provide insights into

Data: Initial parameters: Θ_v^0, local training set: X_v

for $t \in [0, 1, \dots]$ **do**
  /* Local optimization step */
  $\xi_v^t \sim X_v$;
  $\Theta_v^{t+\frac{1}{2}} = \Theta_v^t - \eta \nabla_{\Theta_v^t}(\xi_v^t, \Theta_v^t)$;
  /* Communication with neighbors */
  for $u \in \boldsymbol{N}(v)/\{v\}$ **do**
    send $\Theta_v^{t+\frac{1}{2}}$ to u;
    receive $\Theta_u^{t+\frac{1}{2}}$ from u;
  end
  /* Model updates aggregation */
  $\Theta_v^{t+1} = \frac{1}{|\mathbf{N}(v)|} \sum_{u \in \mathbf{N}(v)} \Theta_u^{t+\frac{1}{2}}$;
end

Algorithm 1: Training protocol for every decentralized user $v \in V$.

the practical challenges of enforcing topological constraints during the deployment of the system.

In this paper, we target our privacy analysis on the D−PSGD protocol proposed by Lian et al. [36]. This protocol provides the same core functionality and properties as the bulk of DL protocols in the literature [1], [14], [25], [30], [34], [35], [38], [55], [66], [70], [74]. Thus, it is representative of the decentralized learning state-of-the-art. Its similarity to FedAVG (see Section 2.1.2) permits a direct comparison between decentralized and federated learning.

In D−PSGD, summarized in Algorithm 1, n users start with common model parameters Θ^0 and iterate over the following three steps until a stop condition is met:

1) *Local training:* Users sample a mini-batch ξ from their (private) local training set and apply gradient descent on their local view of the model parameters. This results in an intermediate model $\Theta_v^{t+\frac{1}{2}}$ that we refer to as *model update*.
2) *Communication:* Users share their model updates $\Theta_v^{t+\frac{1}{2}}$ with their neighbors, and receive their neighbors' updates (line 4 in Algorithm 1).
3) *Aggregation:* Users compute their new model by aggregating *all* their neighbor's updates with their local one. The aggregation is the average of the model parameters.[1]

In contrast to the federated setting, at each round in DL, **users' local set of parameters can be arbitrarily different**. After a suitable number of communication rounds, users find *consensus* on a global state; that is, users' local parameters become equal. In this paper, we measure how close users are from reaching consensus using the *consensus distance* C. This distance is computed as the pairwise discrepancy among local parameters at time t:

$$C(t) = \frac{\sum_{v \in V} \sum_{u \in V/\{v\}} \|\Theta_v^t - \Theta_u^t\|^2}{|V|^2 - |V|}. \quad (1)$$

Intuitively, large values of C indicate that there is a large discrepancy among users' local parameters, whereas small values indicate that users have similar local models. We say that the system has found consensus when C approaches zero.

The D−PSGD protocol does not specify how users select their neighbors nor how they agree on a training setup (including initial parameters of the local models Θ^0). Following the most relevant works in decentralized learning [2], [12], [14], [22], [30], [61], [36], [70], we assume those decisions happened during a honest setup and focus on the fixed communication graph setting; i.e., the graph G does not change over time and users do not drop-off in-between rounds. Nevertheless, our attacks and the result of our analysis apply also to cases in which the topology changes dynamically and users initialization is arbitrary.

We consider three communication topologies:

1. We assume model updates to have equal weights, though, in general, the aggregation of line 8 can be expressed via a weighted sum.

Torus: A regular topology where every user is connected with four other nodes. This topology represents the best-case scenario for decentralized learning given its good mixing properties and higher spectral gap that allow for fast convergence and efficient communication [30]. We consider torus graphs with different number of nodes and we refer to them as *torus-n*, where n is the number of nodes.

Random regular: A regular topology in which all nodes have d random neighbors. Random regular topologies enable us to analyze the impact of the density of connections (d) on the privacy of DL. We refer to these graphs as *regular-(n, d)*, where n is the number of nodes and d the density.

Davis Southern women social network: An unstructured topology which represents a more realistic case, e.g., users communicating in a cross-device setting, used by Koloskova et al [30]. This social network has 32 users with diverse degree. The average degree is 5.74. We refer to this topology as *social-32*.

We provide additional results for different topologies and configurations in Appendix E.

2.1.2. Federated Learning. In Federated Learning (FL) [40] users perform the distributed training process with the support of a central *parameter server* that aggregates and synchronizes model updates among users (Figure 1b). At each iteration, users download the global model from the server and locally apply one or more local training steps. Users send their model updates back to the server. The server aggregates these updates and applies the result to the global parameters, completing a training round.

Compared to DL, there is no direct communication among users in FL. All the communication takes place through the server. Following our notation, we write that every federated user v has neighbors $\mathbf{N}(v) = \{PS\}$, where PS stands for parameter server. As main consequence, users, by design, do not have access to each other's model updates during the training; they can only access the aggregated model update (the average of users' model updates) sent to them by the parameter server. Another consequence of this design is that, at each round, users always share the same set of local parameters for the model f. We refer to those parameters as the *global parameters*.

In our evaluation, we take the Federated Averaging protocol FedAVG [40] as representative of FL algorithms. Motivated by the small amount of users assumed in the decentralized learning literature ($n < 100$) [36], [30], [34], [35], [49], [52], [61], we evaluate a cross-silo federated setting [28], where all users participate in each training round. This setting represents a lower bound for privacy compared to a cross-device federated setting in which only a subset of users participate in every round. We also force users' local training step to be computed on a single, random batch per round to match D−PSGD. Indeed, under this configuration, FedAVG becomes functionally equivalent to D−PSGD where the topology is fully connected (i.e., all users are connected

to each other). This configuration enables a fair comparison between decentralized and federated approaches.

2.1.3. Datasets and architecture. In our experiments, we use the CIFAR-10, CIFAR-100 [33], and STL10 [8] datasets. As in [30], we consider the users' local training sets to be uniformly distributed among users; i.e., every user gets an uniformly sampled (without replacement) fraction $\frac{1}{n}$ of the training set (where $n=|V|$ is the number of users in the system). We use a ResNet20 [21] architecture, with the same hyper-parameters for both the decentralized and federated settings. For each comparison we also consider the same number of users, and the same local training set partition for the decentralized and federated settings. We provide more details about our setup in Appendix C.

2.2. Privacy attacks

We evaluate the privacy offered by decentralized learning using two attacks: *membership inference*, in which an adversary learns whether a target sample is in the training set of a user; and *gradient inversion* in which the adversary can reconstruct samples in the training set of a user.

2.2.1. Membership Inference Attacks. In a Membership Inference Attack (MIA) [56], the adversary tries to infer whether a sample is part, or not, of the training dataset. To make their guess, the adversary can use all information available to them: they can look at the model updates or query the trained model. Vulnerability (or equivalently robustness) to MIA is a good privacy beacon, as membership inference connects to almost all other privacy attacks, e.g., attribute inference attacks can be reduced to MIA [69], [77]. As a result, capturing privacy through MIA is a common choice in the CML literature [57], [69], [37], [56], [68], [5], [43], [42].

In our evaluation, we measure a learning protocol's vulnerability to MIA through the success of a simple metric-based attack. Formally, given a set of model parameters Θ, a local training set X, and a test set $\mathcal{O}$ s.t. $X \cap \mathcal{O} = \emptyset$ and $|X|=|\mathcal{O}|=m$, we estimate membership vulnerability as the accuracy of the membership inference attack over the sets X and $\mathcal{O}$:

$$M(\Theta, X, \mathcal{O}) = \frac{1}{2 \cdot m} \sum_{i=0}^{m-1} [MIA_\Theta(X_i) + \neg MIA_\Theta(\mathcal{O}_i)] \tag{2}$$

$$\text{with} \quad MIA_\Theta(x) = \xi(f_\theta(x)) < \rho, \tag{3}$$

where ξ is the "*label-informed*" entropy [57] and ρ is the optimal threshold. For convenience, when presenting our results we subtract the random guessing baseline (0.5) from the accuracy so that the results we report are centered in 0. We choose to rely on this simple attack because it allows us to quantify the difference in membership vulnerability between DL and FL protocols at a low computational cost (see Section 4). It would be straightforward to run our evaluation with more complex and effective inference attacks including white-box attacks [43], [5], [68]. This would likely increase the vulnerability estimations in DL and FL, but we do not expect that it would significantly affect the difference in between the estimation in each setting.

2.2.2. Gradient inversion. Gradient inversion attacks exploit the observation that the gradient produced by one (or more) SGD steps is just a smooth function of the training data used to compute it. Thus, an attacker capable of accessing users' model updates during a CML protocol may be able to invert them and fully or partially recover the underlying users' private data [16], [26], [71], [79]. The quality of this inversion process is heavily dependent on the configuration used to compute the gradient, the number of trainable parameters of the network and batch size being the most impactful factors.

We adapt two instances of gradient inversion (originally designed for FL) to the DL framework: the passive optimization-based approach proposed by Geiping et al. [16] and the active attack proposed by Boenisch et al. [3]. Our setup seamlessly extends to other attacks [64].

3. The generalization and knowledge trade-off in Decentralized Learning

In this section, we characterize two main byproducts resulting from decentralizing FL. We refer to them as: *local generalization* and adversarial *system knowledge*. We demonstrate that these properties alone prevent honest users' from reaching any meaningful level of privacy in DL.

3.1. Local generalization

Generalization is pivotal to protect the privacy of the training set against attacks based on the model behavior. While well-generalized models may still leak information about the underlying training set [69], [37], it has been demonstrated that poor generalization is the root cause of the privacy risk [56].

Informally, good generalization in CML is achieved when the number of users participating in the learning protocol is maximized: the more users involved in training, the less information about a single individual can be inferred from model updates shared during the protocol.

FL maximizes generalization in this respect: the central server ensures that every state of the global model is computed using *all* the n available model updates, and, importantly, that every model update contributes equally to this computation.

In the decentralized setting, this is not the case. While, as in FL, users' models are a function of the models of all other users in the system; in DL every user has a different "*personalized*" local model to which not all users contribute equally. The contribution of user u_i on user u_j's local parameters depends on the distance between those users in the communication topology. The further these users are, the weaker is the influence of u_i's updates on u_j's model. The

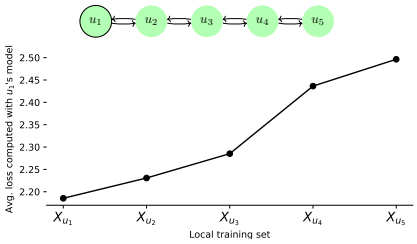


Figure 2: Average loss of u_1's local model computed on every local training set.

strength of the influence decays exponentially with the number of intermediary users, as the updates of u_i get blended with those of intermediary users' models before arriving to u_j due to the gossip-based propagation. We illustrate this phenomenon with a chain-like topology represented in Figure 2 (top). In this topology, u_1 (the first user in the chain) only has one direct neighbor u_2 and the rest of the users in the system are several hops away. Here, user u_1 only receives the updates produced by u_5 after they have been "*consumed*" (i.e., aggregated with other received model updates at the end of a round) and propagated by each intermediate user in the chain within the following model update. When u_5's update reaches u_1, the strength of its signal is reduced by a factor $\frac{1}{|\mathbf{N}(u_4)|} \cdot \frac{1}{|\mathbf{N}(u_3)|} \cdot \frac{1}{|\mathbf{N}(u_2)|} \cdot \frac{1}{|\mathbf{N}(u_1)|} = \frac{1}{54}$ due to the aggregation rule (line 8 Algorithm 1). There is less information about u_5's data compared to model updates produced by closer users (e.g. $\frac{1}{2}$ for u_2). In contrast, in FL every pair of users is virtually separated by a single hop: the server. This ensures that every user's contribution is weighted equally in the global model.

This slow and uneven propagation of updates results in the local model of decentralized users being dominated by their own training set and the training sets of their immediate neighbors, yielding *poor generalization*. We illustrate this effect in Figure 2 (bottom), where we report the average loss when applying u_1's local model on the local training sets of other users. The loss increases with the distance between u_1 and the owner of the local training set. We call this phenomenon "*local generalization*", in contrast to "*global generalization*" offered by federated learning.

After a suitable number of iterations, information will be uniformly propagated in the system and every user's local model will be the same: consensus is met and generalization is maximized. For every round before that, all intermediate user's local models and model updates store more information about their private local training sets than the updates of the other users in the system due to the local generalization phenomenon. As we empirically demonstrate later in the paper, the ability to access these poorly generalized model updates gives a substantial advantage to privacy adversaries compared to what can be learned by just accessing a global (aggregated) model as in FL.

This advantage can only be reduced by limiting the effect of local generalization; that is, by reducing the average distance between each pair of users in the communication graph, or, more pragmatically, increasing the number of neighbors of each node. In the extreme, when all nodes are connected to each other, local effects disappear and the protocol achieves global generalization as in FL.

3.2. System knowledge

Dense topologies, as those needed to limit the effect of local generalization, have negative implications on performance and on privacy. On the performance side, dense topologies increase the communication overhead of users, defeating one of the objectives of decentralized protocols. On the privacy side, dense topologies increase adversaries' knowledge of the state of other users in the system, providing them with more information to perform privacy attacks.

In FL, users can only observe the aggregated result provided by the server, i.e., the global set of parameters. In contrast, in DL, decentralized users receive the model updates of each of their neighbors (Figure 1a). Each of these updates captures a *view of the system*, as it contains information coming from a different subset of nodes in the graph (the neighbors of the neighbors). We call "*system knowledge*" the ability that a user of DL has to access multiple individual model updates per round. The system knowledge gained from having simultaneous access to disjoint views of the system grants an additional advantage to decentralized attackers. As we demonstrate in the next sections, they can combine the model updates they receive and use them to isolate individual contributions of other users, effectively reducing generalization and its positive effect on users' privacy. More critically, we show that an attacker with enough system knowledge can reach the same adversarial capabilities as a parameter server in FL. This effectively defeats another of the objectives of decentralizing learning.

3.3. Take aways

Local generalization and system knowledge are in direct opposition. Increasing the number of users' neighbors as a way to reduce the adversary's advantage coming from local generalization inherently increases the adversary's advantage coming from system knowledge. Conversely, topologies that limit decentralized adversary's system knowledge to prevent them from isolating individual users' contribution inherently increase local generalization, increasing the adversary's advantage.

In the following sections, we quantify the privacy loss stemming from the adversary exploiting this conflict. Our study leads to the conclusion that **the intrinsic trade-off between local generalization and system knowledge fundamentally limits the privacy achievable by users in decentralized learning**.

4. Privacy Against Passive Adversaries

To prove the argument of Section 3, we start by comparing the privacy offered by the DL approach against FL in the

semi-honest model, i.e., when there is a passive adversary in the system. We start by formalizing our threat model.

Passive adversary threat model. A first type of adversaries in CML are passive adversaries. Such adversaries legitimately follow the steps of the CML protocol yet will attempt to learn all possible information from received model updates. Their goal is to infer information about the private training sets of one or more honest users in the system, which we refer to as *victims* or *targets* interchangeably. Passive adversaries do not forge adversarial model updates neither by changing the loss function of the model nor by tampering with their local training set [60].

In this paper, we assume a *weak* passive adversary who has no information about the system (e.g., they know their neighbors but not the rest of the communication topology).

DL passive adversary. Any user involved in a decentralized protocol can be a passive adversary, as they observe the model updates of their neighbors. As the communication topology is always connected in the DL setup [30], [34], [35], every user has at least one neighbor. Thus, DL is required to guarantee privacy against adversarial neighbors as a cardinal property of the protocol. Note that the only scenario that allows decentralized users to rule out adversarial neighbors is when trust is introduced in system; that is, users assume that all their neighbors are fully honest. Current decentralized learning frameworks do not impose any limitation on user connectivity. Thus, an adversarial user can connect to a chosen victim to become their *adversarial neighbor* (see Section 6.3). Hereafter, we denote a passive adversarial user in DL as $\mathcal{A}^{\text{DL}}$.

FL passive adversary. In FL, the server has similar capabilities to an adversarial neighbor in DL, i.e., it receives and sends (an aggregation of) model updates. Federated users can only observe the global model sent by the server at the end of each round. Hereafter, we denote a passive adversarial user in FL as $\mathcal{A}^{\text{FL}}_{\text{user}}$, and a passive adversarial parameter server as $\mathcal{A}^{\text{FL}}_{\text{server}}$.

4.1. Decentralized user vs federated user (passive)

We first compare the privacy that honest users can enjoy against an adversarial decentralized user ($\mathcal{A}^{\text{DL}}$) and an adversarial federated user ($\mathcal{A}^{\text{FL}}_{\text{user}}$). Our results demonstrate that decentralization provides an intrinsic advantage to passive adversarial users compared to the federated setting.

4.1.1. Inference on model updates. As part of any CML protocol, users share the model updates computed on their local data. A passive adversary can use them as parameters for a model f_v and run arbitrary privacy attacks on the victim(s) who generated the updates.

In the DL setting, at every round t of Algorithm 1, the attacker $\mathcal{A}^{\text{DL}}$ receives the model updates $\Theta_v^{t+\frac{1}{2}}$ from each of their neighbors $v \in \mathbf{N}(\mathcal{A}^{\text{DL}})$. In the FL setting, $\mathcal{A}^{\text{FL}}_{\text{user}}$ has only access to the global state of the model provided by the parameter server at the start of the round, which aggregates the updates of all the users in the system.

In Figure 3, we compare the vulnerability of victims against membership inference attacks in DL and FL. We plot the evolution of the vulnerability of the adversary's neighbors to membership inference attacks as the training process progresses. To capture vulnerability, y-axis represents the average MIA accuracy across all the attacker's victims v. From Eq 2, this is computed as:

$$\frac{1}{m}\sum_{v} M(\Theta_v^{t+\frac{1}{2}}, X_v, \mathcal{O}), \tag{4}$$

where $\mathcal{A}$ is either $\mathcal{A}^{\text{DL}}$ (red line) or $\mathcal{A}^{\text{FL}}_{\text{user}}$ (blue line), and the set of victims is either all users in FL (and $m = n$), or the neighbors of $\mathcal{A}$ in DL (and $m = |\mathbf{N}(\mathcal{A})/\mathcal{A}|$). The figure also reports the progression of the consensus distance (Eq. 1) in DL (gray dotted line).

The x-axis aims at capturing the training progression. A natural choice for this axis would be to use the number of protocol iterations t. However, DL and FL do not converge at the same speed. Therefore, DL and FL models at the same round t may be arbitrarily different. To address this limitation, we choose to compare models when they have the same generalization error $g_{err}(t)$ (i.e., same level of *overfitting*). In DL, we compute the average generalization error of users' local parameters at iteration t as:

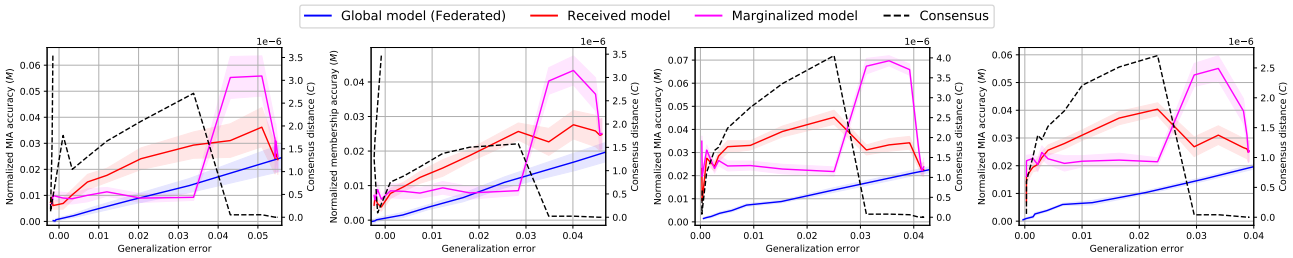
$$g_{err}(t) = \texttt{acc}(X, \Theta^t) - \texttt{acc}(\mathcal{O}, \Theta^t), \tag{5}$$

where, $\Theta^t = \frac{1}{|V|}\sum_{v\in V}\Theta_v^t$ is the average state of the system, $X = \bigcup_{v\in V} X_v$ is the union of all the local training sets, $\mathcal{O}$ is a test set completely disjointed from X, and $\texttt{acc}$ is the accuracy function. In FL, the average state of the system is simply the global model as $\forall_{v\in V}\Theta^t = \Theta_v^t$.

In DL, the vulnerability is a function of both the generalization error and the consensus distance. Larger generalization error denotes overfitting, which is known to result in information leakage about the training set [56]. Larger consensus distance indicates that information is still not uniformly propagated in the system, thus updates carry significantly more information about the local training set than sets from other users. This is what we referred to as *local generalization* in Section 3. We observe that privacy leakage due to local generalization may happen even when the generalization error is close to 0 (leftmost parts of the plots). That is, **even when a decentralized system has perfect generalization, decentralized model updates still contain individualized information that can be used to infer about training data.** In FL, as expected, when assessing the privacy risk at a generalization error level that is near to zero, the privacy risk also approaches zero.

When the DL system reaches consensus (rightmost part of the plots), the vulnerability in DL and FL approaches the same value. This is because when consensus is reached local generalization disappears: decentralized users share the same global model (as in FL). Thus, the victim's model updates do not carry information particular to their own training sets.

The harmful effect of local generalization on privacy can be reduced by increasing the density of the communication



(a) torus-36 & CIFAR-10 (b) social-32 & CIFAR-10 (c) torus-36 & CIFAR-100 (d) social-32 & CIFAR-100

Figure 3: Average MIA vulnerability on four different communication topologies and datasets (DL in red and purple, and FL in blue). For each combination of topology and dataset, we report the average results over 16 runs. In each run, we select a different adversarial user uniformly at random in the system. The halo around the curves reports the standard deviation over the various runs. The gray dotted line represents the consensus distance in DL, computed using Eq. 1.

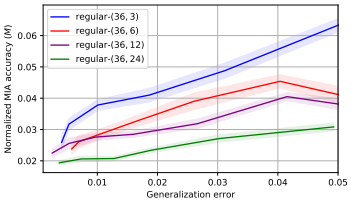


Figure 4: Average MIA vulnerability of model updates generated by decentralized users on regular graphs with increasing density (CIFAR-100).

topology (see Section 3). We confirm this in Figure 4, where we show that increasing the density in random regular topologies (*regular*-$(36, d)$ with $d \in \{3, 6, 12, 24\}$) reduces the privacy harm.

4.1.2. Inference on functionally marginalized model updates. In contrast to $\mathcal{A}^{\text{FL}}_{\text{user}}$, the decentralized adversary $\mathcal{A}^{\text{DL}}$ has access to multiple model updates produced by different users (one per each of their neighbors). This *system knowledge* can be used to further increase membership vulnerability by combining model updates and using them to isolate the local contribution to the update from honest neighbors. To demonstrate this capability, we introduce a novel attack that we call "*functional marginalization*".

Functional marginalization exploits the fact that the local model update Θ^t_v of user v can be divided into two core components:

$$\Theta^t_v \approx \mathring{\Theta}^t_v + \Theta^t_{V/v}, \tag{6}$$

where $\mathring{\Theta}^t_v$ represents the contribution to the update computed with the local training set of the node v, and $\Theta^t_{V/v}$ captures the contributions of all other nodes in the system.

With enough information about $\Theta^t_{V/v}$, and because they know Θ^t_v, the adversary can extract the marginalized contribution $\mathring{\Theta}^t_v$ from Eq 6. Exactly recovering the term $\Theta^t_{V/v}$ is unfeasible. However, the adversary can compute a rough approximation of this value for a victim v from the model updates the adversary receives from other neighbors. The adversary estimates $\Theta^t_{V/v}$ as the average of all parameters they receive, excluding the victim's:

$$\Theta^t_{V/v} = \frac{\sum_{u \in \mathbf{N}(\mathcal{A}^{\text{DL}})/v} \Theta^{t+\frac{1}{2}}_u}{|\mathbf{N}(\mathcal{A}^{\text{DL}})|}. \tag{7}$$

Then, by removing this approximation from the victim's model update, $\mathcal{A}^{\text{DL}}$ isolates the victim's contribution:

$$\mathring{\Theta}^t_v = |\mathbf{N}(\mathcal{A}^{\text{DL}})| \cdot (\Theta^{t+\frac{1}{2}}_v - \Theta^t_{V/v}). \tag{8}$$

This process can also be seen as reversing the aggregation operation in line 8 in Algorithm 1 by pulling out the term $\Theta^{t+\frac{1}{2}}_v$ from the averaged model Θ^{t+1}_v, using somewhat incomplete information.

The recovered "*functionally marginalized model*" $\mathring{\Theta}^t_v$ is a function of the local training set of v only. Thus, the adversary can use it to obtain better results than when attacking directly $\Theta^{t+\frac{1}{2}}_v$, which contains contributions from other users. The difference between the red and purple lines in Figure 3 captures this improvement. As can be seen in the figure, the improvement is not consistent. This is because, as showed in the previous section, membership vulnerability is a function of the global generalization error and the consensus distance. When the consensus distance is high (leftmost part of the plots), $\mathcal{A}^{\text{DL}}$ cannot compute an accurate representation of the global functionality $\Theta^t_{V/v}$. Thus, the marginalized model $\mathring{\Theta}^t_v$ may not be a good representation of the victim's local training set and the attack performs worse than when performed directly on the received model. When the consensus distance C decreases (rightmost part of the plots), $\Theta^t_{V/v}$ becomes a good representation of the global state, the marginalization (Eq. 8) becomes accurate, and membership vulnerability abruptly increases. Finally, when the consensus distance C approaches zero and all users have the same view, and marginalization has no effect as there is no victim's contribution to be isolated. At that point, Eq. 8 results in $\mathring{\Theta}^t_v = \Theta^{t+\frac{1}{2}}_v$ and membership vulnerability is the same as when the received model is attacked directly.

The results in Figure 3 hint that attacks on the received model update or its functionally marginalized version are

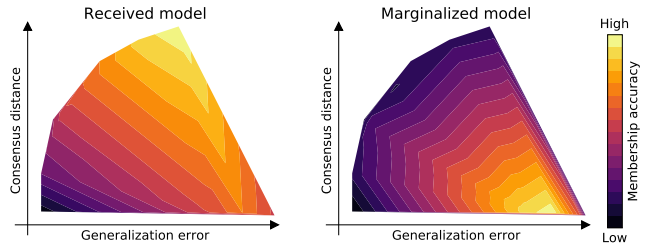


Figure 5: MIA vulnerability as a function of generalization error and consensus distance for the received and marginalized model. Setup: torus-36 on CIFAR-100.

complementary: the former succeeds when the consensus distance is high, the latter succeeds when consensus distance is low. In order to understand the conditions under which attack is more powerful, we compare the two attacks in Figure 5. This figure represents the level of vulnerability (lighter colors represent more vulnerable cases) depending on the generalization error (x-axis) and the consensus distance (y-axis). In both attacks, vulnerability is proportional to the generalization error, but they behave differently depending on the consensus distance. Attacking the received model results in high vulnerability when the distance is large (top center), while when attacking the marginalized model vulnerability is maximized when the consensus distance is low (bottom right). This means that the adversary can maximize their effectiveness by choosing the best attack after evaluating the consensus distance on the received model updates. As before, when the network reaches consensus (i.e., $C(t)=0$ in the rightmost edge of the plots), the vulnerability is minimized as local datasets have the least influence on the updates. In Appendix E.2, we present results obtained on a larger number of users (64 and 128), along with additional topologies. Those show that increasing the number of users magnifies the local generalization phenomenon, which in turn amplifies the vulnerability of model updates. In Appendix E.4, we present an extension of our evaluation to a NLP task, yielding results that are consistent with those in Figure 3.

It should be noted that the inference attacks discussed in this section are *stateless* and can be executed in a single DL round. Therefore, they can be directly applied to dynamic topologies, where the attacker's neighbors may vary during the training process. Additionally, we highlight that the introduced functional marginalization technique is general and can be potentially extended to other CML frameworks. For instance, it can be applied to most Personalized FL frameworks, where users or the server [11], [76] have access to multiple (personalized) versions of users' models.

The results in this section provide empirical support to our claims in Section 3.1; local generalization in DL is an unavoidable source of leakage that does not exist in the FL setup. Ultimately, this means that, for every non-complete topology G (i.e., every topology that induces local generalization), a passive decentralized adversary $\mathcal{A}^{\text{DL}}$ would always be able to infer more information about honest users than an equivalent passive federated adversarial user $\mathcal{A}^{\text{FL}}_{\text{user}}$. In the next section, we show that this claim holds also for any complete topology.

4.2. Decentralized user vs federated server (pass.)

We now compare the adversarial capabilities of an adversarial passive decentralized user $\mathcal{A}^{\text{DL}}$, against an adversarial passive federated server $\mathcal{A}^{\text{FL}}_{\text{server}}$. Our results demonstrate that an adversarial user in DL can have the same adversarial capabilities as a parameter server in FL.

In FL, the adversary $\mathcal{A}^{\text{FL}}_{\text{server}}$ is in a privileged position to run privacy attacks. Unlike adversarial federated users $\mathcal{A}^{\text{FL}}_{\text{user}}$, who only receive aggregate model updates, the parameter server has access to user's model updates and the intermediate states of their local optimization processes – pseudo-gradients for `FedAVG`. This position enables $\mathcal{A}^{\text{FL}}_{\text{server}}$ to perform powerful privacy attacks such as accurate inference attacks on gradients [46] or gradient inversion [16], [26], [48], [71], [78], [79].

To carry out a gradient inversion attack, an attacker $\mathcal{A}$ needs two pieces of information: (1) the gradient $\nabla_{\Theta_v^t}\mathbf{L}(\xi_v^t)$, and (2) the parameters of the network Θ_v^t used to compute such a gradient. These two components are, by design, available to an adversarial parameter server in FL. However, they are not directly accessible to an adversarial user in DL $\mathcal{A}^{\text{DL}}$.

An attacker $\mathcal{A}^{\text{DL}}$ in DL receives the following model update $\Theta_v^{t+\frac{1}{2}}$ from their neighbor v:

$$\Theta_v^{t+\frac{1}{2}} = \Theta_v^t - \eta\nabla_{\Theta_v^t}\mathbf{L}(\xi_v^t). \tag{9}$$

To extract the gradient $\nabla_{\Theta_v^t}\mathbf{L}(\xi_v^t)$ from this model update, $\mathcal{A}^{\text{DL}}$ needs to know Θ_v^t. In principle, the exact value Θ_v^t is not available to the attacker as it is a function of the model updates from v's neighbors. In principle, this could render gradient inversion attacks in DL unfeasible.

4.2.1. Gradient inversion attack in decentralized learning. We now show how the adversary $\mathcal{A}^{\text{DL}}$ can estimate the gradient $\nabla_{\Theta_v^t}\mathbf{L}(\xi_v^t)$ in order to perform the gradient inversion attack.

There are three ways in which $\mathcal{A}^{\text{DL}}$ can perfectly recover the individual gradient of their neighbors. The first two are trivial cases in which $\Theta_v^t = \Theta^t_{\mathcal{A}^{\text{DL}}}$: the first training iteration $t=0$, and when users $\mathcal{A}^{\text{DL}}$ and v achieve consensus (i.e., $C(t)=0$). In both cases, the attacker can recover the victim's gradient by computing $\frac{1}{\eta}(\Theta_v^{t+\frac{1}{2}} - \Theta^t_{\mathcal{A}^{\text{DL}}})$.

Gradient recovery at $t=0$ could be prevented by having users choose different initial parameters Θ^0, with the caveat that this modification may impact the learning process. The second case, however, cannot be avoided. Reaching consensus is the goal of decentralized learning. Thus, eventually, the attacker will have the opportunity to recover the gradient and perform the inversion attack.

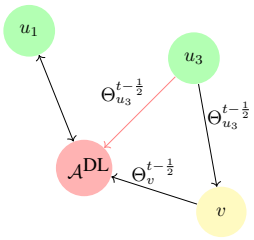


Figure 6: $\mathcal{A}^{\text{DL}}$ is able to access the gradient of the victim node v because of its connection with the honest node u_3.

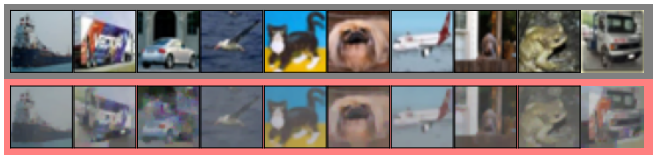


Figure 7: Examples of reconstruction (red panels) obtained via gradient inversion on the node v (see Figure 6) using [16] for a batch size of size 16 on the CIFAR-10 dataset.

Gradient recovery from system knowledge. The third situation in which the adversary $\mathcal{A}^{\text{DL}}$ can recover neighbors' gradients is when the set of attacker's neighbors $\mathbf{N}(\mathcal{A}^{\text{DL}})$ is a super-set of the victim's neighbors set $\mathbf{N}(v)$. This situation, provides $\mathcal{A}^{\text{DL}}$ with enough system knowledge to recover the gradient regardless of the time step t.

In order to recover the gradient produced by v, an attacker $\mathcal{A}^{\text{DL}}$ needs to subtract the unknown set of parameters Θ_v^t from the received model update $\Theta_v^{t+\frac{1}{2}}$. Recall that the victim's set of parameters Θ_v^t is defined as:

$$\Theta_v^t = \frac{1}{|\mathbf{N}(v)|} \sum_{u \in \mathbf{N}(v)} \Theta_u^{t-\frac{1}{2}}, \tag{10}$$

where, $\Theta_u^{t-\frac{1}{2}}$ is the model broadcasted by the node u to all its neighbors during the previous training iteration $(t-1)$. If the adversary has access to the updates of the neighbors of the victim, they can use these neighbours updates at time $t-1$ to reconstruct $\sum_{u \in \mathbf{N}(v)} \Theta_u^{t-\frac{1}{2}}$, and use this to compute Θ_v^t.

Figure 6 illustrates this issue. The local model of the victim v (in yellow) is:

$$\Theta_v^t = \frac{1}{3}(\Theta_v^{t-\frac{1}{2}} + \Theta_{\mathcal{A}^{\text{DL}}}^{t-\frac{1}{2}} + \Theta_{u_3}^{t-\frac{1}{2}}), \tag{11}$$

where $\Theta_{u_3}^{t-\frac{1}{2}}$ is a model update produced by another user who is not under the control of the attacker (i.e., u_3). If the attacker $\mathcal{A}^{\text{DL}}$ (in red) also has access to u_3's updates (i.e., $\mathbf{N}(v) \subset \mathbf{N}(\mathcal{A}^{\text{DL}})$), then they can recompute the local state Θ_v^t as in Eq. 11 and recover the gradient signal from v's model updates as $\nabla_{\Theta_v^t} = \Theta_v^{t-\frac{1}{2}} - \Theta_v^t$.

Once $\mathcal{A}^{\text{DL}}$ has $\nabla_{\Theta_v^t}$, they can run an arbitrary inversion attack. Figure 7 shows a sample of images reconstructed via gradient inversion for the topology in Figure 6 obtained using the optimization-based method proposed in [16]. Given the same underlying setup (e.g., same batch size), the result of the inversion attack is equivalent to the one that would be achieved by $\mathcal{A}^{\text{FL}}_{\text{server}}$. Note that this is not a targeted attack and $\mathcal{A}^{\text{DL}}$ can perform gradient recovery on all the neighbors for which the condition is met simultaneously at every given round.

The neighbors-discovery trick. A condition for $\mathcal{A}^{\text{DL}}$ to be able to perform the gradient recovery attack described above is that they must know the exact set of neighbors of their target v: $\mathbf{N}(v)$. We now show how, even if the attacker does not know the global communication topology, they can learn $\mathbf{N}(v)$ from the model updates whenever $\mathbf{N}(v) \subseteq \mathbf{N}(\mathcal{A}^{\text{DL}})$.

In a nutshell, the neighbors-discovery trick finds the model updates of the previous round that explain the victim's model update received at the current time step. More formally, it searches for the set $Q \subseteq \mathbf{N}(\mathcal{A}^{\text{DL}})$ such that $E(Q) = 0$, where E is defined as:

$$E(Q) = \Theta_v^{t+\frac{1}{2}} - (\tilde{\Theta}_Q + \tilde{\nabla}_Q) \quad \text{with} \tag{12}$$

$$\tilde{\Theta}_Q = \frac{1}{|Q|} \sum_{u \in Q} \Theta_u^{t-\frac{1}{2}} \quad \text{and} \quad \tilde{\nabla}_Q = \Theta_v^{t+\frac{1}{2}} - \tilde{\Theta}_Q \tag{13}$$

When $Q = \mathbf{N}(v)$, we have $\tilde{\Theta}_Q + \tilde{\nabla}_Q = \Theta_v^{t+\frac{1}{2}}$ and the subtraction in Eq. 12 results in 0. When there is no $Q \subseteq \mathbf{N}(\mathcal{A}^{\text{DL}})$ s.t. $E(Q) = 0$, the attacker learns that currently they are not connected to all the victim's neighbors. In Appendix B, we empirically demonstrate the effectiveness of this discovery trick. We note that Eq. 12 is linear and can be solved via linear/dynamic programming. The chances of successfully discovering the neighbors—and consequently the ability to perform gradient inversion—is maximized when the adversary has as many neighbors as possible.

It is worth noting that techniques like gradient recovery and the neighbor discovery trick are effective even in a dynamic communication topology setting, where users select their neighbors on a round-by-round basis. The attacker only needs to be connected to the victim for a minimum of two consecutive rounds to execute such attacks.

Summing up, **a passive adversarial user in decentralized learning can be as powerful as a passive server in the federated setup**. As we show, it is the case for all victims v such that $\mathbf{N}(v) \subseteq \mathbf{N}(\mathcal{A}^{\text{DL}})$. Since DL allows the adversary to connect to users of their choice, hence to be connected to all users, $\mathcal{A}^{\text{DL}}$ eventually is as powerful as $\mathcal{A}^{\text{FL}}_{\text{server}}$: just like an adversarial server, $\mathcal{A}^{\text{DL}}$ can (1) observe the model update of every user in the system and (2) isolate the individual gradients of a user. It is also trivially true when the DL topology is fully connected to begin with.

4.3. Take aways

In Section 4.1, we show that an adversarial decentralized user can exploit the local generalization of any non-complete topology to launch membership inference attacks. To limit this leakage, the density of the communication topology

must increase, up to the complete topology, where there is no local generalization phenomenon anymore. Increasing connectivity, however, is in conflict with the conclusions of Section 4.2 which show that increasing connectivity increases the system knowledge of the adversary. Giving the adversary the ability to collect additional information on the system, results in even more significant leakage, and enables powerful attacks such as gradient inversion.

We conclude that there is no topology G for which DL provides better, or even equal, honest user protection against a passive adversary in the network than the one federated users enjoy. In other words, the adversary $\mathcal{A}^{\text{DL}}$ is more powerful than $\mathcal{A}^{\text{FL}}_{\text{user}}$ *regardless* of the underlying DL setup. In addition, $\mathcal{A}^{\text{DL}}$ can acquire the same adversarial capabilities as $\mathcal{A}^{\text{FL}}_{\text{server}}$, as long as there are no constraints on how users connect to each other in DL (or the topology is complete). As consequence, while in FL there is at most one powerful adversary: the server, in DL there may be multiple powerful adversaries: any user with enough connections. This means that contrary to what is claimed by its proponents, DL does not reduce the capabilities of adversaries. Rather **the power of adversaries, and so the privacy vulnerability of honest users, is multiplied**.

5. Privacy Against Active Adversaries

In this section, we compare the privacy offered by the DL approach against the FL alternative in the malicious model, i.e., when there is an active adversary in the system. As in Section 4, we start by formalizing our threat model.

Active adversary threat model. Active adversaries in CML (eq. malicious) behave maliciously during the protocol execution. In this paper, we instantiate such an adversary by allowing the them to send arbitrary model updates to their neighbors in addition to their passive capabilities. We refer to an active adversarial user in DL as $\bar{\mathcal{A}}^{\text{DL}}$, and to a malicious user and a malicious parameter server in FL as $\bar{\mathcal{A}}^{\text{FL}}_{\text{user}}$ and $\bar{\mathcal{A}}^{\text{FL}}_{\text{server}}$, respectively.

5.1. Decentralized user vs federated user (active)

In CML, the effectiveness of active attacks is proportional to the capability of adversarial users of influencing the model parameters of their victim [24], [42], [43]. Intuitively, this is because the adversary uses the victim model updates as input for making inferences. These updates are a function of the victim's local model parameters and training set. By influencing the victim's model parameters, the adversary can modify the model updates to leak more information about the private training set [3], [15], [24], [43], [50], [64].

In both FL and DL, a user computes their local model parameters v as the aggregation of their own model update and the model updates of other users in the system:

$$\Theta_v^{t+1} = \frac{1}{m}\Theta_v^{t+\frac{1}{2}} + \frac{1}{m}\Theta_{u_1}^{t+\frac{1}{2}} + \cdots + \frac{1}{m}\Theta_{u_{m-1}}^{t+\frac{1}{2}}, \qquad (14)$$

where m is the number of users participating in the aggregation (all users in FL, and the neighbors of v in DL).

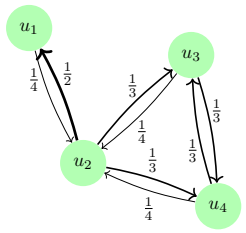


Figure 8: Figure describing the **direct** influence factor (edge thickness and label) of each node on neighbors's local models for a given topology composed of four users.

Assuming that the model updates are bounded in norm, any user can influence at most a fraction $\frac{1}{m}$ of v's model. The larger m is, the smaller the influence a single adversarial user can have on v's model. In the federated setting, by definition all users participate in the aggregation ($m = n$). Thus, the influence an adversarial user can have on their victim is the minimum possible. In decentralized settings, the number of neighbors, and thus the level of adversarial influence, depends on the topology (see Figure 8). For the DL settings that offer a significant cost advantage with respect to FL, the topology is sparse and therefore users have a small number of neighbors ($m << n$). In such scenario an adversarial user $\bar{\mathcal{A}}^{\text{DL}}$ in DL always has higher **influence** over their targets' model parameters than an equivalent malicious user $\bar{\mathcal{A}}^{\text{FL}}_{\text{user}}$ in FL. The influence is only equal to an FL adversarial user when the DL topology is fully connected, but at that point there is no advantage in decentralizing the learning process.

Besides the increase of influence on each honest user, decentralization enables active adversarial users $\bar{\mathcal{A}}^{\text{DL}}$ to send m different updates to their m neighbors each round. This is in contrast to the FL scenario where $\bar{\mathcal{A}}^{\text{FL}}_{\text{user}}$ can only submit a single model update per round (to the server). This extra capability enable adversarial users in DL to carry out attacks that an adversarial FL user cannot launch.

We briefly introduce the "*echo attack*", as example of an active attack that can be performed by $\bar{\mathcal{A}}^{\text{DL}}$ but not by $\bar{\mathcal{A}}^{\text{FL}}_{\text{user}}$. (More details can be found in Appendix A.) During an echo attack, the adversary 'echoes' back each received model update (or a variation, e.g., the functionality marginalized version of the model update) to the neighbor who sent it, in order to push them to overfit on their local training data. Our experimental results show a significant increase in the generalization error ($4\times$), and consequently in the privacy leakage, when performing the echo attack (see Figures 11 and 12 in Appendix A). The federated adversary $\bar{\mathcal{A}}^{\text{FL}}_{\text{user}}$, which only receives the aggregated model from the server, cannot isolate users' individual contributions and is left with one possibility: echoing the global model to the server, which will have little to no effect on the generalization error of users. In general, decentralized users can perform any active attack in reach for federated users (e.g. [24], [42], [43]), while the converse is not true.

Adversarial influence, like local generalization, grows with sparsity of the underlying topology. Therefore, simi-

larly to local generalization, it can be diminished by increasing the number of neighbors m of the victim. When users are connected to all nodes in the system, m is maximized and their models reach global generalization, e.g., node u_2 in Figure 8.

In conclusion, given that all the attacks that $\bar{\mathcal{A}}^{\text{FL}}_{\text{user}}$ can perform can be also performed by $\bar{\mathcal{A}}^{\text{DL}}$ and that $\bar{\mathcal{A}}^{\text{DL}}$ has always greater influence on victims than $\bar{\mathcal{A}}^{\text{FL}}_{\text{user}}$, **any non-complete topology in DL offers less protection against active privacy attacks from malicious users than the equivalent federated approach**. In Section 5.2.1, we show that this result extends to the complete topology.

5.2. Decentralized user vs federated server (active)

We now compare the adversarial capabilities of active adversarial users ($\bar{\mathcal{A}}^{\text{DL}}$) in decentralized learning against an active adversarial parameter server in FL ($\bar{\mathcal{A}}^{\text{FL}}_{\text{server}}$).

A malicious parameter server $\bar{\mathcal{A}}^{\text{FL}}_{\text{server}}$ is the strongest active attacker possible: it can arbitrarily decide the local state of any user in one single iteration. This results in extremely effective privacy attacks [3], [15], [50], [64]. These attacks cannot be directly applied by $\bar{\mathcal{A}}^{\text{DL}}$, as obtaining such influence on victims' models is inconceivable in the decentralized setting, regardless of the underlying topology. Indeed, even when the attacker is the only neighbor of the victim, their influence on the victim's model is in theory at most $\frac{1}{2}$, since the victim aggregates the adversary's contribution with their own local information (see edge (u_2, u_1) in Figure 8). However, we show how $\bar{\mathcal{A}}^{\text{DL}}$ can use system knowledge to achieve full influence on a victim's state, just like a malicious federated server.

5.2.1. State-override attack. In Section 4.2.1, we show that system knowledge enables the attacker to remove the effect of generalization and isolate victims' gradients. This knowledge can also be used to cancel out contributions coming from honest neighbors on the victim's aggregated model. The goal of the adversary is not to isolate information, but to increase its influence capability. We now introduce the "*state-override attack*", in which the adversary uses this capability to *override* the result of the local model aggregation computed by the victim at line 8 of Algorithm 1.

Formally, given a target v and an adversary $\mathcal{A}$ such that $\mathbf{N}(v) \subseteq \mathbf{N}(\mathcal{A})$, the adversary can distribute the following model update to override the victim's model with parameters $\tilde{\Theta}$ chosen by the adversary:

$$\Theta_{\mathcal{A}}^{t+\frac{1}{2}} = -\Big(\sum_{u \in \mathbf{N}(v)/\mathcal{A}} \Theta_u^{t+\frac{1}{2}}\Big) + |\mathbf{N}(v)| \cdot \tilde{\Theta}. \tag{15}$$

This forged update contains the negated, partial aggregation in Eq. 15 of the model updates from the victim's neighbors $\mathbf{N}(v)/\mathcal{A}$.

Figure 9: Examples of reconstruction obtained via state-override attack and gradient inversion (malicious initialization [3]) on the node v (see Figure 6) for a batch size of size 64 on the STL10 dataset.

Upon receiving the model updates, the victim v proceeds to aggregate the receive inputs locally:

$$\Theta_v^{t+1} = \frac{1}{|\mathbf{N}(v)|} \sum_{u \in \mathbf{N}(v)} \Theta_u^{t+\frac{1}{2}} = \frac{\big(\sum_{u \in \mathbf{N}(v)/\mathcal{A}} \Theta_u^{t+\frac{1}{2}}\big) + \Theta_{\mathcal{A}}^{t+\frac{1}{2}}\big]}{|\mathbf{N}(v)|} = \tilde{\Theta}. \tag{16}$$

The adversary's update cancels the contribution of the neighbors, and the result of the aggregation becomes the "*payload*" $\tilde{\Theta}$.

With this attack, the adversary can take complete control of the victim's parameters regardless the number of the victim's neighbors. As a result, the adversary can perform attacks such as [3], [15], [50], [64] within two iterations: one to override the model and one to extract the result. This is equivalent to a single round of federated learning.

To give a concrete example of the impact of the state-override attack on the privacy of users in DL, we consider active gradient inversion attacks. Boenisch et al demonstrate in [3] that the effect of gradient inversion can be greatly magnified in both reconstruction quality and applicability when the attacker has full control on the parameters used to compute the gradient. The attacker can inject the victim's network with maliciously crafted parameters that force the computed gradient to artificially memorize more information than intended about the input batches [15], [64], [3]. The state-override attack is the perfect way for the adversary to get control on the parameters used to compute the gradient.

In our setting, the attacker $\bar{\mathcal{A}}^{\text{DL}}$ first uses the *state-override attack* to maliciously force the victim v's local state to be the adversary-chosen parameters $\tilde{\Theta}$ created according to [3]. In the next round, $\bar{\mathcal{A}}^{\text{DL}}$ first receives v's model update. Second, recovers the gradient signal as $\nabla_{\tilde{\Theta}}\mathbf{L}(\xi_v^t) = \Theta_v^{t+\frac{1}{2}} - \tilde{\Theta}$. And third performs the inversion. We show the results of this attack in Figure 9. It is important to note that the adversary $\mathcal{A}$ can, at each round, simultaneously perform the state-override attack on all users whose neighbors are a subset of the adversary's neighbors. It suffices to send a different adversarial model update (computed according to Eq.15) to each neighbor within the same communication round, and perform the gradient inversion steps discussed above in parallel.

To perform the state-override attack as described above, $\mathcal{A}$ must be a *rushing* adversary, i.e., the last user to

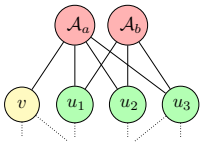


Figure 10: Minimal example of secure aggregation evasion for an aggregation threshold of 3 users.

communicate its model update, so as to know the inputs of the victim's neighbors beforehand. If the adversary cannot do this (e.g., if the system has a broadcast schedule), $\mathcal{A}$ can use model updates from the previous round and achieve comparable results, as we show in Appendix D.

Additionally, while in this work we focus on privacy attacks, the state-override attack can also be used as a stepping stone towards robustness attacks. Trivially, it enables the attacker to plant arbitrarily backdoor/trojan functionality in users' models [18] or completely destroy the utility of their models (e.g., by setting the payload to random parameters).

5.3. Take aways

We show that active adversaries can gain full influence over honest users' state. This enables them to mount privacy attacks with extremely high quality of reconstruction of the users' training sets. Like in the semi-honest setting, active adversarial users in decentralized learning can be as powerful as a malicious parameter server as long as the underlying topology allows for it. When the attacker is fully-connected, they can arbitrarily decide the parameters of **all the honest users in the system every round**, matching the functionality of the FL server. In practice, this means that, with respect to current real-world deployments of federated learning [19], [41], [67] in which the parameter server must be (semi-)honest in order to guarantee a meaningful level of privacy to users [3], [15], [50], the decentralized learning paradigm increases the number of potential adversaries, and thus the number of entities that need to be (semi-)honest for users to have privacy.

6. Defences

In this section, we discuss the suitability of different defense techniques from the CML literature and their effectiveness to prevent the attacks we introduce in this the paper.

6.1. Secure Aggregation

A way to prevent system-knowledge-based attacks is to use secure aggregation (SA) protocols [4]. When using SA, users privately perform the aggregation step (line 8 of Algorithm 1) without revealing their model updates in the clear to each other. Users can only access the result of the aggregation after their update has been averaged with those of other users. Since individual model updates are not observable by the adversary, SA can eliminate attacks relying on system knowledge such as gradient recovery, functional marginalization, or state-override.[2]

Therefore, by using SA and a fully-connected topology to achieve global generalization, DL offers users the same level of privacy as federated learning against passive adversaries (see Section 3). However, this setting would impose a significant overhead: every decentralized user has the same communication complexity as a parameter server in FL, and in addition the overhead imposed by cryptographic operations needed for secure aggregation. This overhead with respect to FL would come at no gain in privacy.

Evading SA in Decentralized Learning Even if the overhead introduced by the protocol would be acceptable, securely implementing SA in decentralized learning poses a significant challenge. We demonstrate below how an attacker can always retrieve the model update of another user if they can impersonate or compromise an additional node in the system.

Essentially, the attacker can obtain the model update of a victim v by calculating the difference between two aggregated values that differ only by v's model update. More formally, given $\mathcal{A}_a$ and $\mathcal{A}_b$ the nodes under the control of the attacker $\mathcal{A}$ and a victim node v, $\mathcal{A}$ can recover the victim's model update Θ_v^t, by choosing $\mathbf{N}(\mathcal{A}_b) = \mathbf{N}(\mathcal{A}_a)/v$. Once the attacker nodes received the aggregated values, these can recover v's model update by computing: $\Theta_v^t = \text{SA}(\sum_u^{\mathbf{N}(\mathcal{A}_a)}) - \text{SA}(\sum_u^{\mathbf{N}(\mathcal{A}_b)})$. An example of this configuration is depicted in Figure 10. This approach does not require any auxiliary knowledge on the victim, and $\mathbf{N}(\mathcal{A}_a)$ can be chosen arbitrarily by the attacker. We remark that this simple SA-evasion technique is independent from the employed aggregation protocol and they would work even under verifiable SA or SA performed via Trusted Execution Environment (TEE) [47]. Assuming fault resilient SA [4] (which is necessary under real-world deployments), this strategy would work also in a complete topology, where $\mathbf{N}(\mathcal{A}_a) = \mathbf{N}(\mathcal{A}_b)$. In this case, it is enough for $\mathcal{A}_b$ to simulate the drop-off of the victim. In the general case, this technique would be remain applicable as long as the threshold for SA is greater equal to $|\mathbf{N}(\mathcal{A}_b)| - 1$.

More research is needed to find effective and reliable topology-aware SA in decentralized learning.

Differential privacy. A formal approach to achieve privacy in DL would be using Differential Privacy (DP) [13], such as differential-private SGD to implement the local optimization steps.

In decentralized learning, the lack of a trusted, centralized curator (role taken by the parameter server in federated learning) prevents the use of central-DP. Thus, DL protocols have to resort to local-DP. Local-DP results in a worse trade-off between privacy and utility compared to central-DP [27], [45]. One common way to improve this trade-off is to use distributed-DP [6], [27] which assumes the existence of an effective secure aggregation protocol that would only

2. Although gradient recovery would unavoidably succeed when $\mathbf{N}(v) = \{\mathcal{A}\}$ (always) or $\mathbf{N}(\mathcal{A}) = \{v\}$ (when $C = 0$).

reveal the noisy sum of the local model updates to the aggregator. Distributed-DP allows to tune the local noise proportionally to the number of users m participating at the aggregation ($\sim \frac{1}{m}$). As for plain secure aggregation (see Section 6.1), the success of this approach depends on the density of the topology. The lower the number of neighbors of a user, the less participants in the aggregation, and the more noise users have to add locally to achieve a desired level of privacy. Increasing the number of neighbors would solve this issue, but would also increase the communication overhead, suppressing the advantage of decentralized learning over the federated approach. Indeed, as for SA, distributed-DP matches the utility of federated learning only when the topology is complete.

In summary, decentralized learning cannot match the utility/privacy trade-off of the federated setting given existing DP techniques due to the lack of a centralized curator and the need to keep its communication overhead advantage. This gap may be reduced using differentially-private techniques tailored to decentralized learning. The community already started moving in this direction [7], [9], [65], so far achieving only limited results.

Finally, while these perturbation-based defenses may work, they can be also applied in FL. Thus, they do not result on any privacy advantage for the decentralized setting. In fact, techniques such as distributed-DP [6], [27] (which uses SA as a primitive) are easier, and more efficient, to apply to FL protocols compared to DL.

6.2. Robust aggregation protocols

Robust aggregation methods [29] aim at reducing the influence of active adversaries on the local state of users by replacing the plain average-based aggregation (line 8 of Algorithm 1) with more robust metrics. These techniques can neither prevent the privacy attacks we propose nor confer any advantage to decentralization.

Robust aggregation techniques trade privacy for robustness as they rely on magnifying the influence of local information (the current state of the user) over external one (model updates provided by other users). This amplifies the local generalization effect, increasing the information our attacks can exploit. Robust aggregation can *hamper* attacks such as the state-override attack, but not prevent them entirely. We demonstrate this is the case in Appendix A.1, where we apply our attacks to self-centering clipping [22]—a robust aggregation protocol for DL.

Furthermore, robust aggregation can also be applied to the federated setting by letting federated users maintain a consistent local state and implement any user-side robust aggregation. Therefore, they do not provide an advantage for DL with respect to the federated approach.

6.3. Constraining the communication topology

Decentralized learning advocates often point out that freedom to choose neighbors is a positive and unique feature of decentralization. In this paper, we thoroughly demonstrate that such freedom can be leveraged by the adversary to boost their capabilities to the point of achieving the same attack power as the parameter server in federated learning.

To address this issue, the communication topology underlying DL should be carefully designed if we wish to prevent certain attacks. This means that systems in which users join the network without constraints are unworkable, as individual decisions are unlikely to match any pre-defined topology. In fact, it is actually hard to enforce constraints without a central orchestrator that has a global knowledge of the system as highlighted by years of research on peer-to-peer anonymous communications [17], [44], [54], [53], [63]. Yet, introducing such a powerful central entity in the system would result on new security threats if this entity is malicious: Assuming a malicious central orchestrator who can arbitrary choose the communication topology is equivalent to assuming a malicious parameter server in FL. Trivially, the orchestrator can maliciously design the topology in order to grant full adversarial capability to itself (and carry the attacks in Sections 4.2 and 5.2).

n siItuations where trust can been established among users within the system, the ability to choose neighbors can also have a positive impact. If users have the ability to differentiate between trustworthy and untrustworthy nodes, they can selectively connect with users whom they trust to be honest and reject connection requests from those who are untrusted. This approach can lead to secure configurations without the need for strict topological constraints, as it assumes that no honest user would connect to the untrusted adversarial nodes. However, this approach requires making strong assumptions about both the behavior of users and the security of the underlying implementation.

7. Conclusion

Fully decentralized collaborative machine learning has been proposed as a potential solution for preserving user privacy while avoiding the performance issues associated with federated learning. In this work, we introduced a series of attacks that show that existing decentralized learning protocols do not deliver on their promised privacy properties. Our results indicate that decentralized users possess adversarial capabilities that are comparable to those of a federated parameter server.

We also show that due to the increase in capabilities decentralization confers to adversaries, existing defenses cannot prevent all the attacks we propose. In order for decentralized learning to provide the same level of privacy guarantees as federated learning, it must give up on any potential performance gains. We hope that our findings can serve as benchmarks for the research community, inspiring the development of new design principles that enable truly privacy-preserving decentralized learning.

Acknowledgements

This work was partially supported by *Fondation Botnar*.

References

[1] A. Bellet, R. Guerraoui, M. Taziki, and M. Tommasi, "Personalized and private peer-to-peer machine learning," 2018.

[2] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, and J. Stainer, "Machine learning with adversaries: Byzantine tolerant gradient descent," in *NeurIPS*, 2017.

[3] F. Boenisch, A. Dziedzic, R. Schuster, A. S. Shamsabadi, I. Shumailov, and N. Papernot, "When the curious abandon honesty: Federated learning is not private," *arXiv preprint*, 2021.

[4] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for privacy-preserving machine learning," in *CCS*. ACM, 2017.

[5] N. Carlini, S. Chien, M. Nasr, S. Song, A. Terzis, and F. Tramèr, "Membership inference attacks from first principles," in *IEEE S&P*, 2022.

[6] W.-N. Chen, C. A. Choquette-Choo, P. Kairouz, and A. T. Suresh, "The fundamental price of secure aggregation in differentially private federated learning," *arXiv preprint*, 2022.

[7] H.-P. Cheng, P. Yu, H. Hu, S. Zawad, F. Yan, S. Li, H. Li, and Y. Chen, "Towards decentralized deep learning with differential privacy," in *Cloud Computing – CLOUD*. Springer-Verlag, 2019.

[8] A. Coates, A. Ng, and H. Lee, "An analysis of single-layer networks in unsupervised feature learning," in *AISTATS*. PMLR, 2011.

[9] E. Cyffers and A. Bellet, "Privacy amplification by decentralization," *arXiv preprint*, 2020.

[10] E. Cyffers, M. Even, A. Bellet, and L. Massoulié, "Muffliato: Peer-to-peer privacy amplification for decentralized optimization and averaging," in *Advances in Neural Information Processing Systems*, A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, Eds., 2022. [Online]. Available: https://openreview.net/forum?id=QotmVXC-8T

[11] R. Dai, L. Shen, F. He, X. Tian, and D. Tao, "DisPFL: Towards communication-efficient personalized federated learning via decentralized sparse training," in *ICML*. PMLR, 2022.

[12] G. Damaskinos, E. M. E. Mhamdi, R. Guerraoui, R. Patra, and M. Taziki, "Asynchronous byzantine machine learning (the case of SGD)," in *ICML*. PMLR, 2018.

[13] C. Dwork, "Differential privacy," in *ICALP (2)*. Springer, 2006.

[14] E.-M. El-Mhamdi, S. Farhadkhani, R. Guerraoui, A. Guirguis, L. N. Hoang, and S. Rouault, "Collaborative learning in the jungle," *arXiv preprint*, 2020.

[15] L. H. Fowl, J. Geiping, W. Czaja, M. Goldblum, and T. Goldstein, "Robbing the fed: Directly obtaining private data in federated learning with modified models," in *International Conference on Learning Representations*, 2022.

[16] J. Geiping, H. Bauermeister, H. Dröge, and M. Moeller, "Inverting gradients - how easy is it to break privacy in federated learning?" in *NeurIPS*, 2020.

[17] S. Goel, M. Robson, M. Polte, and E. G. Sirer, "Herbivore: A Scalable and Efficient Protocol for Anonymous Communication," Tech. Rep., 2003.

[18] T. Gu, B. Dolan-Gavitt, and S. Garg, "Badnets: Identifying vulnerabilities in the machine learning model supply chain," *arXiv preprint*, 2017.

[19] A. Hard, K. Rao, R. Mathews, S. Ramaswamy, F. Beaufays, S. Augenstein, H. Eichner, C. Kiddon, and D. Ramage, "Federated learning for mobile keyboard prediction," 2019.

[20] C. He, C. Tan, H. Tang, S. Qiu, and J. Liu, "Central server free federated learning over single-sided trust social networks," *arXiv preprint*, 2019.

[21] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*. IEEE, 2016.

[22] L. He, S. P. Karimireddy, and M. Jaggi, "Byzantine-robust decentralized learning via self-centered clipping," *arXiv preprint*, 2022.

[23] I. Hegedűs, G. Danner, and M. Jelasity, "Gossip learning as a decentralized alternative to federated learning," in *IFIP International Conference on Distributed Applications and Interoperable Systems*, 2019.

[24] B. Hitaj, G. Ateniese, and F. Perez-Cruz, "Deep models under the gan: Information leakage from collaborative deep learning," in *CCS*. ACM, 2017.

[25] C. Hu, J. Jiang, and Z. Wang, "Decentralized federated learning: A segmented gossip approach," *arXiv preprint*, 2019.

[26] J. Jeon, J. Kim, K. Lee, S. Oh, and J. Ok, "Gradient inversion with generative image prior," in *NeurIPS*, 2021.

[27] P. Kairouz, Z. Liu, and T. Steinke, "The distributed discrete gaussian mechanism for federated learning with secure aggregation," in *ICML*. PMLR, 2021.

[28] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. A. Bonawitz, Z. Charles, G. Cormode, R. Cummings, R. G. L. D'Oliveira, S. E. Rouayheb, D. Evans, J. Gardner, Z. Garrett, A. Gascón, B. Ghazi, P. B. Gibbons, M. Gruteser, Z. Harchaoui, C. He, L. He, Z. Huo, B. Hutchinson, J. Hsu, M. Jaggi, T. Javidi, G. Joshi, M. Khodak, J. Konečný, A. Korolova, F. Koushanfar, S. Koyejo, T. Lepoint, Y. Liu, P. Mittal, M. Mohri, R. Nock, A. Özgür, R. Pagh, M. Raykova, H. Qi, D. Ramage, R. Raskar, D. Song, W. Song, S. U. Stich, Z. Sun, A. T. Suresh, F. Tramèr, P. Vepakomma, J. Wang, L. Xiong, Z. Xu, Q. Yang, F. X. Yu, H. Yu, and S. Zhao, "Advances and open problems in federated learning," *CoRR*, 2019.

[29] S. P. Karimireddy, L. He, and M. Jaggi, "Learning from history for byzantine robust optimization," in *ICML*. PMLR, 2021.

[30] A. Koloskova, T. Lin, S. U. Stich, and M. Jaggi, "Decentralized deep learning with arbitrary communication compression," in *International Conference on Learning Representations*, 2020.

[31] A. Koloskova, N. Loizou, S. Boreiri, M. Jaggi, and S. Stich, "A unified theory of decentralized SGD with changing topology and local updates," in *ICML*. PMLR, 2020.

[32] A. Koloskova, S. Stich, and M. Jaggi, "Decentralized stochastic optimization and gossip algorithms with compressed communication," in *ICML*. PMLR, 2019.

[33] A. Krizhevsky, "Learning multiple layers of features from tiny images," Tech. Rep., 2009.

[34] A. Lalitha, O. C. Kilinc, T. Javidi, and F. Koushanfar, "Peer-to-peer federated learning on graphs," *arXiv preprint*, 2019.

[35] A. Lalitha, S. Shekhar, T. Javidi, and F. Koushanfar, "Fully decentralized federated learning," in *Third workshop on Bayesian Deep Learning (NeurIPS)*, 2018.

[36] X. Lian, C. Zhang, H. Zhang, C.-J. Hsieh, W. Zhang, and J. Liu, "Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent," in *NeurIPS*, 2017.

[37] Y. Long, V. Bindschaedler, L. Wang, D. Bu, X. Wang, H. Tang, C. A. Gunter, and K. Chen, "Understanding membership inferences on well-generalized learning models," *arXiv preprint*, 2018.

[38] S. Lu, Y. Zhang, and Y. Wang, "Decentralized federated learning for electronic health records," in *2020 54th Annual Conference on Information Sciences and Systems (CISS)*, 2020.

[39] O. Marfoq, G. Neglia, A. Bellet, L. Kameni, and R. Vidal, "Federated multi-task learning under a mixture of distributions," in *NeurIPS*, 2021.

[40] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*. PMLR, 2017.

[41] H. B. McMahan, D. Ramage, K. Talwar, and L. Zhang, "Learning differentially private recurrent language models," *arXiv preprint*, 2017.

[42] L. Melis, C. Song, E. De Cristofaro, and V. Shmatikov, "Exploiting unintended feature leakage in collaborative learning," in *IEEE S&P*, 2019.

[43] N. Milad, S. Reza, and H. Amir, "Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning," in *IEEE S&P*, 2019.

[44] P. Mittal and N. Borisov, "Shadowwalker: peer-to-peer anonymous communication using redundant structured topologies," in *CCS*. ACM, 2009.

[45] M. Naseri, J. Hayes, and E. De Cristofaro, "Local and central differential privacy for robustness and privacy in federated learning," *arXiv preprint*, 2020.

[46] M. Nasr, R. Shokri, and A. Houmansadr, "Comprehensive privacy analysis of deep learning," in *IEEE S&P*, 2018.

[47] J. Nguyen, K. Malik, H. Zhan, A. Yousefpour, M. Rabbat, M. Malek, and D. Huba, "Federated learning with buffered asynchronous aggregation," *arXiv preprint*, 2021.

[48] X. Pan, M. Zhang, Y. Yan, J. Zhu, and M. Yang, "Theory-oriented deep leakage from gradients via linear equation solver," 2020.

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

[50] D. Pasquini, D. Francati, and G. Ateniese, "Eluding secure aggregation in federated learning via model inconsistency," 2021.

[51] Y. Pei, R. Mao, Y. Liu, C. Chen, S. Xu, F. Qiang, and B. E. Tech, "Decentralized federated graph neural networks," in *FTL-IJCAI*, 2021.

[52] A. G. Roy, S. Siddiqui, S. Pölsterl, N. Navab, and C. Wachinger, "Braintorrent: A peer-to-peer environment for decentralized federated learning," *arXiv preprint*, 2019.

[53] M. Schuchard, A. W. Dean, V. Heorhiadi, N. Hopper, and Y. Kim, "Balancing the shadows," in *WPES*. ACM, 2010.

[54] F. Shirazi, M. Simeonovski, M. R. Asghar, M. Backes, and C. Díaz, "A survey on routing in anonymous communication protocols," *ACM Comput. Surv.*, 2018.

[55] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *2015 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, 2015.

[56] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *IEEE S&P*, 2017.

[57] L. Song and P. Mittal, "Systematic evaluation of privacy risks of machine learning models," in *USENIX Security Symposium*, 2021.

[58] T. Sun, D. Li, and B. Wang, "Decentralized federated averaging," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[59] TensorFlow, "Text classification with movie reviews," https://www.tensorflow.org/text/tutorials/text_classification_rnn, 2023, accessed: 2023-03-23.

[60] F. Tramèr, R. Shokri, A. S. Joaquin, H. Le, M. Jagielski, S. Hong, and N. Carlini, "Truth serum: Poisoning machine learning models to reveal their secrets," 2022.

[61] T. Vogels, L. He, A. Koloskova, S. P. Karimireddy, T. Lin, S. U. Stich, and M. Jaggi, "Relaysum for decentralized deep learning on heterogeneous data," in *NeurIPS*, 2021.

[62] J. Wang, A. K. Sahu, Z. Yang, G. Joshi, and S. Kar, "Matcha: Speeding up decentralized sgd via matching decomposition sampling," in *2019 Sixth Indian Control Conference (ICC)*, 2019.

[63] Q. Wang, P. Mittal, and N. Borisov, "In search of an anonymous and secure lookup: attacks on structured peer-to-peer anonymous communication systems," in *CCS*. ACM, 2010.

[64] Y. Wen, J. A. Geiping, L. Fowl, M. Goldblum, and T. Goldstein, "Fishing for user data in large-batch federated learning via gradient magnification," in *ICML*, K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato, Eds. PMLR, 2022.

[65] H. Xiao, Y. Ye, and S. Devadas, "Local differential privacy in decentralized optimization," *arXiv preprint*, 2019.

[66] H. Xing, O. Simeone, and S. Bi, "Decentralized federated learning via sgd over wireless d2d networks," in *IEEE 21st International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, 2020.

[67] 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," *arXiv preprint*, 2018.

[68] J. Ye, A. Maddi, S. K. Murakonda, and R. Shokri, "Enhanced membership inference attacks against machine learning models," *arXiv preprint*, 2021.

[69] S. Yeom, I. Giacomelli, M. Fredrikson, and S. Jha, "Privacy risk in machine learning: Analyzing the connection to overfitting," in *CSF*. IEEE, 2018.

[70] D. Yin, Y. Chen, K. Ramchandran, and P. L. Bartlett, "Byzantine-robust distributed learning: Towards optimal statistical rates," *arXiv preprint*, 2018.

[71] H. Yin, A. Mallya, A. Vahdat, J. M. Alvarez, J. Kautz, and P. Molchanov, "See through gradients: Image batch recovery via gradinversion," in *CVPR*. IEEE, 2021.

[72] B. Ying, K. Yuan, Y. Chen, H. Hu, P. Pan, and W. Yin, "Exponential graph is provably efficient for decentralized deep training," *NeurIPS*, 2021.

[73] B. Ying, K. Yuan, H. Hu, Y. Chen, and W. Yin, "Bluefog: Make decentralized algorithms practical for optimization and deep learning," *arXiv preprint*, 2021.

[74] Y. Yuan, R. Chen, C. Sun, M. Wang, F. Hua, X. Yi, T. Yang, and J. Liu, "Defed: A principled decentralized and privacy-preserving federated learning algorithm," 2021.

[75] S. Zehtabi, S. Hosseinalipour, and C. G. Brinton, "Event-triggered decentralized federated learning over resource-constrained edge devices," 2022.

[76] M. Zhang, K. Sapra, S. Fidler, S. Yeung, and J. M. Alvarez, "Personalized federated learning with first order model optimization," in *International Conference on Learning Representations*, 2021.

[77] B. Z. H. Zhao, A. Agrawal, C. Coburn, H. J. Asghar, R. Bhaskar, M. A. Kaafar, D. Webb, and P. Dickinson, "On the (in)feasibility of attribute inference attacks on machine learning models," in *IEEE EuroS&P*, 2021.

[78] J. Zhu and M. B. Blaschko, "R-GAP: recursive gradient attack on privacy," *arXiv preprint*, 2020.

[79] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," in *NeurIPS*, 2019.

Appendix A. Echo attacks

Echo attacks exploit the high influence factor of a decentralized attacker and system knowledge to force the local state of a chosen neighbor victim to leak information about its private training set. The aim of the attacker is driving the victim's local model towards severe "*overfitting*", forcing it to memorize the local training set

beyond what would be memorized in a honest execution. This amplifies the information leakage stemming from local generalization, worsening the impact of the attacks presented in the previous sections.

To carry out an echo attack, at the broadcast step, instead of their own model, the adversary $\bar{\mathcal{A}}^{\text{DL}}$ broadcasts echos of the victim's updates at the current or previous round. Such *echo* updates artificially increase the relevance of the victim's local training set in the victim's local model. This diminishes the influence of other users in the system on the victim's model, magnifying the effect of local generalization. More formally, during the echo attack, the adversary starts by collecting the neighbors' model updates and uses them to craft the echo update $\tilde{\Theta}$. In particular, the attacker uses the functionally marginalized version (Section 4.1.2) of the victim model update (i.e., $\tilde{\Theta}=\mathring{\Theta}_v^t$) which approximately captures the isolated contribution of the victim. The adversary broadcasts $\tilde{\Theta}$ to all the attacker's neighbors including the victim, increasing overfitting at the next step of the victim's honest execution (line 8 Algorithm 1). The effect of echo attacks is magnified by the iterative interaction between the attacker and the victim and by the "*echo chamber effect*" that results from the neighbors of the attacker also propagating the malicious echo update to the victim via second order interactions. We formalize echo attacks in Algorithm 2.

Echo attacks are extremely efficient and easy to carry: the adversary does not require a local training set or any information on the learning task and they have very low computational cost, as it does not need to train a local model but only post-process the received updates. Note that malicious FL users ($\bar{\mathcal{A}}^{\text{FL}}_{\text{user}}$) cannot replicate echo attacks as they cannot isolate other users' individual contributions from the observable global state of the system and, thus, they cannot broadcast echo updates.

While conceptually simple, echo attacks are extremely effective in practice. Figure 11 compares the generalization error of the victim's model against other non-target users in the system at different training iterations during the echo attack. While this gap is larger at the start of the training, on average, the generalization error of the target is about 10 times more than the non-targets'. As seen in the previous section, the increase in the generalization error creates a massive privacy risk for the target node.

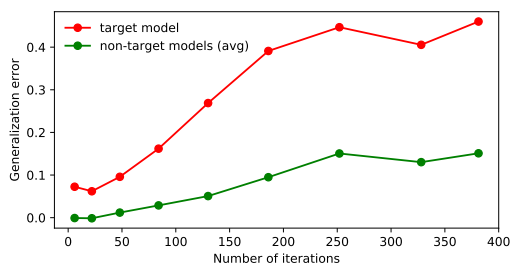


Figure 11: Generalization errors for the target user of the echo attack and the non-target ones during the protocol for the setup: torus-36, ResNet20 and CIFAR-100.

We show the effectiveness of the echo attack in Figure 12 on various configurations (green lines) and compare them to the results obtained with the passive inference attacks of Section 4.1 (dashed lines). Even when the system finds consensus, the privacy risk for the target remains high. This is because the attacker's echo updates have actively influenced the global state of the system (not only the victims's one) by artificially increasing the relevance of the victims's contribution. For the *social-32* topologies, we observe a large standard deviation. This is because the impact of the attack depends on the connectivity of the victim. Recall that the strength of an active attack is proportional to the influence factor of the attacker, which is inversely proportional to the number of neighbors of the victim (see Section 5.1). We illustrate this phenomenon in Figure 13, where we evaluate the effect of the echo attack on targets with different number of neighbors on regular graphs with an increasing density. We keep the degree of the attacker fixed to 3 in order to isolate the impact of the victim's connectivity on the privacy risk. We see that, as we predicted in Section 5.1, low degree boosts the impact of active attacks on users.

Also, attackers can improve their effectiveness by choosing their position in the communication topology to maximize their influence on the system. Like for gradient inversion, the best strategy is to maximize their number of neighbors. If this is not possible, attackers can also aim to be in a position that maximizes the closeness centrality (or other centrality metrics) with the victim to strengthen the "*echo chamber effect*". However, adversaries can only use this strategy if they know the global topology. Finally, we note that if the attacker has the victim as sole neighbor or the marginalized model cannot be computed, the adversarial model update can be set to $\tilde{\Theta}=\Theta_v^{t+\frac{1}{2}}$ (i.e., victim's model update), obtaining inferior but comparable performance; we show this in Figure 15 in Appendix E.

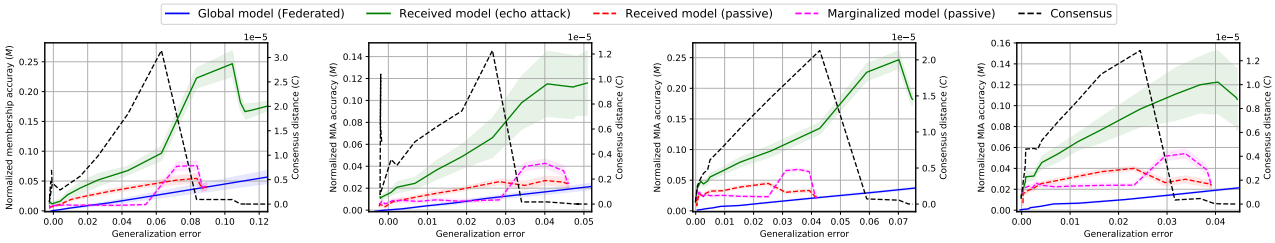
A.1. Echo attack on robust aggregation.

One common approach to reduce the adversarial influence of active attackers in both the federated and decentralized setting is to use robust aggregation methods [29]. An example for the decentralized setup is the work of He et al. [22]. This work proposes to hamper the influence of byzantine nodes by using self-centered clipping. Nodes clip the received model updates in the τ-sphere around their current local model before aggregating them:

$$\Theta_v^{t+1}=\sum_{u\in\mathbf{N}(v)}\left[w_{i,j}\cdot(\Theta_v^{t+\frac{1}{2}}+\texttt{CLIP}(\Theta_u^{t+\frac{1}{2}}-\Theta_v^{t+\frac{1}{2}},\tau))\right], \tag{17}$$

where $\texttt{CLIP}(x,\tau)=\min(1,\tau/||x||)\cdot x$.

This approach hides a trade-off between generalization and robustness. The clipping procedure simply degrades the information provided by the other users in the system in favor of the local one. This successfully reduces the effectiveness of general active attacks. However, it also reduces the generalization of the users' local models, magnifying the harmful effect of local generalization. Because there is



(a) torus-36 & CIFAR-10 (b) social-32 & CIFAR-10 (c) torus-36 & CIFAR-100 (d) social-32 & CIFAR-100

Figure 12: Average MIA vulnerability during an *echo* attack, on four different combinations of communication topologies and training sets for DL and FL.

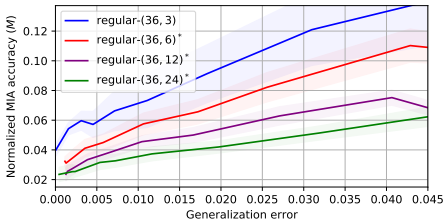


Figure 13: Effect of different numbers of neighbors for the target of the echo attack using CIFAR-100 as training set.

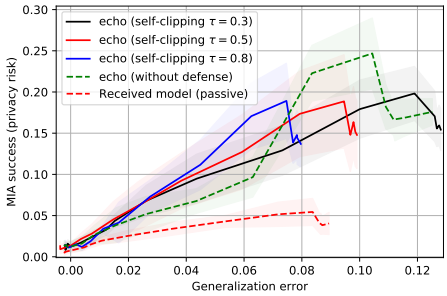


Figure 14: Effect of the self-centered clipping robust aggregation on the echo attack for *torus-36* and *CIFAR-10*.

less information from others, the local model updates retain more information about the local training set of the user.

Eventually, self-centered clipping produces a very similar effect than an echo attack: the influence of local parameters is magnified. Therefore, this defense tends to amplify our attacks rather then defending from them. We show this effect in Figure 14, where we compare the performance of echo attacks on systems with and without self-centered-clipping [22]. Of course, when τ gets closer to 0, the system degenerates to non-collaborative learning (every node trains its model locally). In this case, active attacks such as echo does not offer much inference advantage to the adversary.

Appendix B.
Neighbors discovery trick

Under deployment, weights are computed in finite precision. Then, due to floating-point arithmetic, Eq. 12 does not always result in a precise 0. For the practical case, it is enough to search for:

$$\underset{Q \in \mathbf{N}(A^{\mathrm{DL}})}{arg\,min} \; |\Theta_v^{t+\frac{1}{2}} - (\tilde{\Theta}_Q + \tilde{\nabla}_Q)|, \tag{18}$$

obtaining almost perfect accuracy (see Eq. 12). To validate this claim, we use the torus-16 topology, the worst-case for the adversary given its intrinsic regularity. The attacker

Input: victim node: v

for $t \in [0, 1, \dots]$ **do**

  `/* Receive model updates from neighbors */`

  for $u \in N(\mathcal{A})/\{\mathcal{A}\}$ **do**

    receive $\Theta_u^{t+\frac{1}{2}}$ from u;

  end

  `/* Forge adversarial model update */`

  $\tilde{\Theta} = \mathring{\Theta}_v^t = (|\mathbf{N}(\mathcal{A})|-1)(\Theta_v^{t+\frac{1}{2}} - \frac{\sum_{u \in \mathbf{N}(\mathcal{A})/\{v,\mathcal{A}\}} \Theta_u^{t+\frac{1}{2}}}{|\mathbf{N}(\mathcal{A})|-1})$;

  `/* Broadcast the malicious model update */`

  for $u \in N(\mathcal{A})/\{\mathcal{A}\}$ **do**

    send $\tilde{\Theta}$ to u;

  end

end

Algorithm 2: *echo* attack for an active attacker node $\mathcal{A}$.

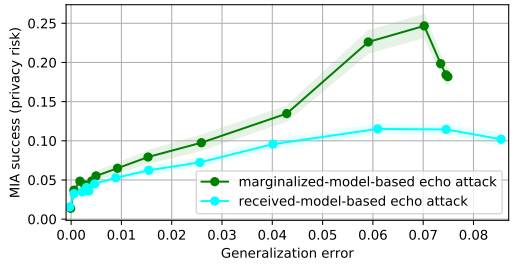


Figure 15: Comparison between echo attack based on marginalized model (green) and received model (cyan) for CIFAR-100, *torus-36* and ResNet20.

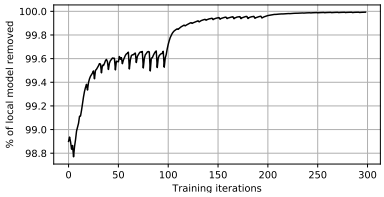
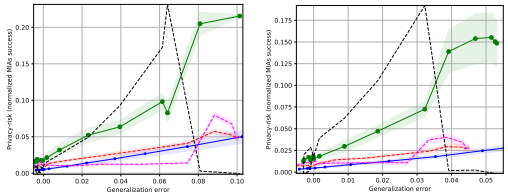


Figure 16: Performance of the state-override attack with inexact information for the topology in Figure 6.



(a) torus-36 & CIFAR-10 (b) social-32 & CIFAR-10

Figure 17: Average MIA vulnerability on two combinations of communication topologies and training sets for DL and FL when using a shallow CNN architecture.

is fully connected and wants to enumerate the local connections of all the other users in the system. We train a ResNet20 architecture on CIFAR-10 for 10 rounds. We then use the model updates received by the attacker to perform the neighbors discovery trick using Eq. 18. We repeat the experiment 32 times. Eq. 18 finds the exact set of neighbors 98.7% of the time.

Yet, it is possible that due to finite precision arithmetic a subset $\dot{Q} \neq \mathbf{N}(v)$ such that $E(\dot{Q}) \leq E(\mathbf{N}(v))$ exists (see Eq. 12). To reduce this probability, it is enough to run Eq. 18 when the consensus distance is high, which can be triggered by a malicious user through model inconsistencies, or to compute Eq. 18 over different rounds.

Appendix C. Details on model training

- **Training set partition:** The training set is uniformly partitioned among users. Given a training set X: every user gets a disjointed sample from X of size $\frac{|X|}{n}$, where n is number of users in the system. No data augmentation is performed.
- **Optimizer:** We use SGD with momentum ($\alpha = 0.9$).
- **Learning rate:** We anneal the learning rate during the training to speed up consensus for the decentralized systems. The initial learning rate is set to 0.1, then we scale it by 0.1 at iterations $200, 350$ and 450 during the training. We do not schedule the learning for federated learning.
- **Batch size:** 256.
- **Stop condition:** We train the models with *early-stopping*. We stop the training when the accuracy of the average of the local models on the validation set stops improving (with a patience of 3).

Our code will be available upon publication.

Appendix D. State-override attack with inexact information

When the users are forced to send their updates synchronously in the decentralized protocol, an attacker will not receive the model updates of their neighbor before sending their own model update. To perform the state-override attack, the adversary then needs to rely on the model updates of the previous round. Of course, this results in an inexact suppression of the current state of the victim. We show the result of the state-override attack using model updates received at the previous round in Figure 16. At worst, the attacker controls 98.7% of the local state of the target. It approaches 100% at the end of the training. We conclude that the state-override attack is effective even when adversaries cannot choose when to send their updates.

Appendix E. Additional results

E.1. Shallower architectures

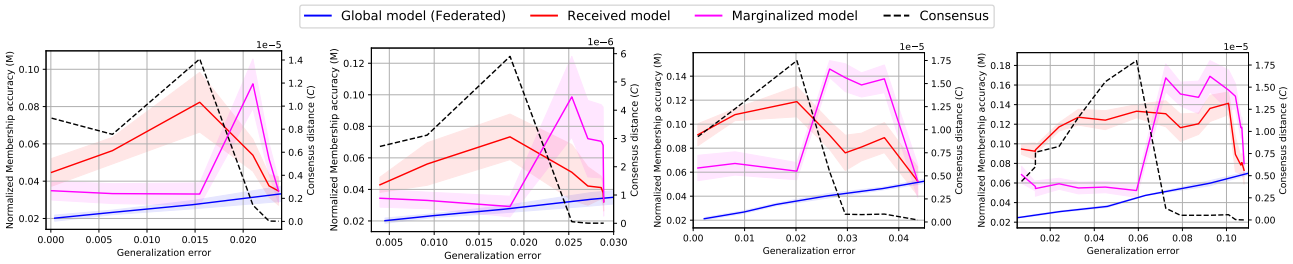
In Figure 17 we report the MIA accuracy for a shallow Convolution Neural Network (CNN) of $225,000$ parameters for both the passive and active attacks. While the privacy risk for the received model tends to be lower, the attacks match what was observed with the deeper ResNet20 model.

E.2. Scaling-up number of users

In Figure 20 we report the MIA vulnerability of the model updates and their marginalized version in the *torus-64* topology. Comparing these results to the *torus-36* topology (Figure 3), it is evident that the sparsity of the topology augments the vulnerability of the received model updates. As the number of users augments (but their number of neighbors stays 4), the average distance between users increases, which boosts the local generalization phenomenon and the inherent privacy risk associated (Section 3). This result reinforces our observation: sparse topologies reduce individual user's privacy. To keep privacy risk constant, the topology density must adapt whenever new users join the protocol. We observe similar results for *torus-128* and *expander-128* in Figure 18, where *expander-128* is an *Erdős–Rényi* random graph with edge probability $\frac{log(n)}{n}$ as defined in [10].

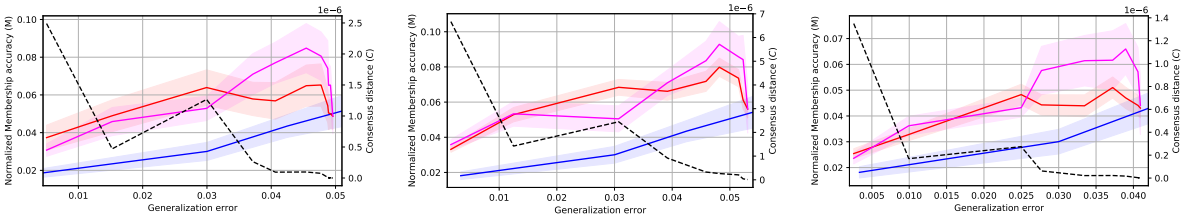
E.3. Multiple local iterations

We further investigate the effect of local generalization on different DL setups. In particular, we consider the impact of multiple optimization steps in the local optimization phase of clients (line 3 of Algorithm 1). Results are reported



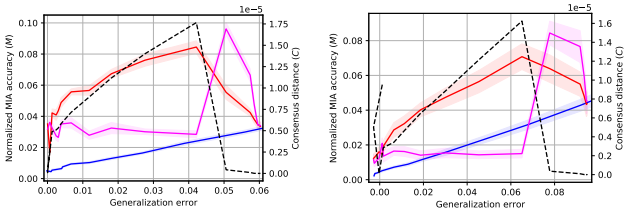
(a) torus-128 & CIFAR-10 (b) expander-128 & CIFAR-10 (c) torus-128 & CIFAR-100 (d) expander-128 & CIFAR-100

Figure 18: Average MIA vulnerability on four different combinations of communication topologies and datasets (DL in red and purple, and FL in blue). For each combination, we report the average results over 16 runs.



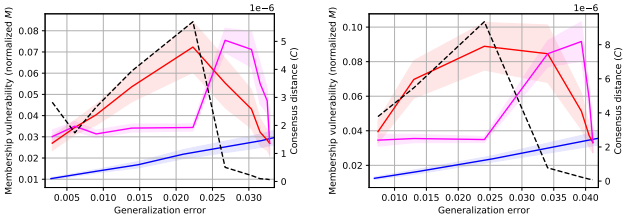
(a) torus-36 & imdb-reviews (b) expander-36 & imdb-reviews (c) social-32 & imdb-reviews

Figure 19: Average MIA vulnerability on three DL setups on a text classification task.



(a) torus-64 & CIFAR-10 (b) torus-64 & CIFAR-100

Figure 20: Average MIA vulnerability on two different combinations of communication topologies and training sets for distributed and federated learning (ResNet20 architecture).



(a) torus-36; CIFAR-10; 2 steps (b) torus-36; CIFAR-10; 3 steps

Figure 21: Average MIA vulnerability on two DL setups with different number of local optimization steps.

in Figure 21, where 2 and 3 steps are considered for *torus-36* and *CIFAR-10*. Compared to the single step optimization setting (Figure 3), running multiple optimization rounds boosts the local generalization phenomenon, magnifying the leakage inherent in the model updates. This effect is particularly marked for the MIAs based on the shared model updates (red lines).

E.4. Different input domains

In this section, we briefly explore data domains beyond the realm of vision, which is the primary focus of the decentralized learning literature. Specifically, we assess the effect of local generalization on a Natural Language Processing (NLP) task. To conduct our experiments, we utilize the widely-used *imdb-reviews* dataset, which involves a movie review classification task. The experimental setup we employ is the one described in [59]. In particular, in this setting, our aim is to assess the ability of an attacker to infer the membership of an entire review (i.e., a sequence of words) in the local training set of other nodes that are part of the collaborative training process. In Figure 19, we report results for the topologies *torus-36*, *social-32*, and *expander-36*. As evident from the results, although models exhibit varying behaviors due to the different learning task, the effectiveness of the inference attacks described in Section 4.1 is consistent with the one observed in the vision domain.