Data Poisoning Attacks in Gossip Learning

Alexandre Pham, Maria Potop-Butucaru, Sébastien Tixeuil, Serge Fdida

Data Poisoning Attacks in Gossip Learning

Alexandre Pham*, Maria Potop-Butucaru*, Sébastien Tixeuil*°, and Serge Fdida*

Abstract Traditional machine learning systems were designed in a centralized manner. In such designs, the central entity maintains both the machine learning model and the data used to adjust the model's parameters. As data centralization yields privacy issues, Federated Learning was introduced to reduce data sharing and have a central server coordinate the learning of multiple devices. While Federated Learning is more decentralized, it still relies on a central entity that may fail or be subject to attacks, provoking the failure of the whole system. Then, Decentralized Federated Learning removes the need for a central server entirely, letting participating processes handle the coordination of the model construction. This distributed control urges studying the possibility of malicious attacks by the participants themselves. While poisoning attacks on Federated Learning have been extensively studied, their effects in Decentralized Federated Learning did not get the same level of attention. Our work is the first to propose a methodology to assess poisoning attacks in Decentralized Federated Learning in both churn free and churn prone scenarios. Furthermore, in order to evaluate our methodology on a case study representative for gossip learning we extended the gossipy simulator with an attack injector module.

Key words: Gossip learning, Decentralized federated learning, Poisoning attacks, Methodology

1 Introduction

Machine learning algorithms use statistical methods to make predictions or classifications, and to discover patterns and insights in data. Typically, a ML approach consists in collecting data, selecting a model, and tuning model parameters based on collected data (in the model training phase) before actually using the trained model.

In traditional systems relying on ML, a central entity manages both the ML model and the data collected for training the model. Such data centralization is problematic with regard to risk and responsibilities [13] when the data used to train the model is sensitive and should be kept private, or simply to obey regulations.

In 2016, Google introduced Federated Learning (FL) [13] as a solution to privacy-wise Machine Learning. In this framework, the central entity only manages a global model and coordinates the training across local devices (e.g. smartphone,

* Sorbonne Université, CNRS, LIP6, F-75005 Paris, France e-mail: Name.Surname@lip6.fr
° Institut Universitaire de France, Paris, France

Acknowledgements: The work presented in this document has received funding from the EU Horizon Europe research and innovation Programme under Grant Agreement No. 101070118.

laptop etc.), which use their own data (that they do not share). Then, devices send their newly adjusted local model's parameters to the central entity that aggregates it with other local models to create a new global model, and this process repeats. This method enables private collaborative learning. However, despite its reduced role, the existence of a central server yields a signle point of failure, and an obvious attack target, as summarized by Liu et al. [11], and by Xia et al. [19].

Decentralized Federated Learning (DFL) aims to do FL without relying on a central server [2], using *P2P* or *Gossip Communications*. The former relies on an existing architecture to operate, while the latter assumes direct communications in a neighborhood. In DFL, FL based attacks and defenses have also been studied. For example, Bernstein et al. [3, 4] with *SignSGD* in the FL context, where they firstly present and study their work as a compression mechanism, and then study it as a defense mechanism. Later, Qu et al. adapted *SignSGD* in the DFL context [16]. In the following, we focus on Gossip Learning (GL), which recently was instrumental in many applications such as building a recommendation system [1], or improving Channel State Information feedback performance [8].

Ormándi et al. [14] is one of the eldest traces of GL. Each node periodically sends their model to a neighbor. When a node receives a model, the node merges it with its current model and considers the result as its new model. Later, Hegedűs et al. [9] improved the work by Ormandi et al [14] by introducing two useful mechanisms in communication restrained networks. After, Danner et al. [5] improved the merging process used by Hegedűs et al. [9]. It should be noted that none of the aforementioned papers address attack or defense in this context.

In this work, we assess the impact of attackers (also called *Byzantine* or *malicious*) nodes that try to poison models build through a GL algorithm. Our benchmark GL algorithm was proposed by Hegedűs et al. [9] as best-in-class in this line of research. Close to our work, Giaretta and Girdzijauskas [7] studied the applicability of GL, highlighting problems and providing fixes when data distribution is correlated to either degree distribution or communication speed. However, they do not consider Byzantine nodes in their work.

Our contributions. Our contribution is threefold. First, we propose a methodology to assess the effects of a poisoning attack in a system that executes a gossip learning algorithm. Second, in order to evaluate our methodology, we implemented an extension of the popular *gossipy*[1] simulator. Our extensions are publicly available (with execution details)[2], and implement a poison injector in *gossipy*, which allows to assess the performances of both clean and corrupted dataset simultaneously. Finally, we apply our methodology on the state of the art gossip learning algorithm by Hegedűs et al [9]. Our findings show that the resilience of this algorithm to poisoning attacks depends on several factors such as topology, Byzantine nodes distribution, and churn. Our analysis pave the way to efficiently design countermeasures against poisoning attacks. We organize this work as follows. Section 2 briefly describes our case study. In Section 3 we describe our methodology. We

[1] `https://github.com/makgyver/gossipy/`

[2] `https://gitlab.lip6.fr/apham/data-poisoning-attacks-in-gossip-learning`

present our results based on this methodology in Section 4. We discuss and conclude with Section 5.

2 Case study: State-of-the-art Gossip Learning

Gossip Learning is a way to do Federated Learning in decentralized systems without a central coordinator via gossip exchanges of parameters or updates, directly between nodes. Recently, Hegedűs et al [9] proposed a GL algorithm that outperforms FL [13] with respect to the global performances (in the ML perspective) when taking into account communication resources. Their algorithm, *Partitioned Token Gossip Learning Algorithm (PTGLA)* uses two interesting mechanisms for communication-restrained networks such as IoTs. The first one is a compression mechanism to decrease message size, by not sending all parameters during each exchange, but part of it called Partition and, we denote S the total number of partitions that the system is using, a high S implies smaller messages. The second mechanism enables us to control the flow of communications, thanks to *Tokens*, by achieving a balance between sending messages *proactively*, i.e. periodically, which may lead to communication flooding and sending messages *reactively*, i.e. after an event e.g. a local update or a message received, which may lead to starvation (no message circulates in the network).

In the sequel, we study the resilience of this algorithm to various poisoning attacks.

3 Methodology

In this section, we present the various elements of our methodology: the choice of the topologies, Dataset and ML algorithm, the attack, the churn settings, and the metrics used to assess the impact of the attacks.

Topologies for GL infrastructure. We investigate the following topologies with n nodes:

- 20 fan-out network, as originally used by Hegedűs et al [9];
- random 20-regular with bidirectional links and same number of links for every node;
- Watts-Strogatz ($k = 20, p = 0.5$) for its small-world property [17];
- Erdős-Rényi $\left(\text{with } p = \frac{2\log(n)}{n}\right)$ for its balanced property [6];
- Zipf law graphs, where each node has at least one neighbor, with $\alpha = 2$.

For the last two topologies, we restrict our simulations to instances where the network is connected.

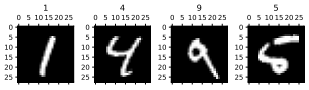
Dataset for ML model training. In this work, we use the MNIST handwritten digit database [10] as our dataset. It is made of 2 sets: a training set and a test set that we

denote $\mathscr{D}_{\text{training}}$ and $\mathscr{D}_{\text{test}}$, of cardinal 60000 and 10000 respectively[3]. An instance of this dataset (either from the training or test) is a couple $(x,y) \in \mathbb{R}^{784} \times [\![0,9]\!]$, x represents a 28×28 pixel images with the number y written on it. Each node k will have as a local training set $\mathscr{D}^k_{\text{training}}$ of cardinal 250, which is drawn from $\mathscr{D}_{\text{training}}$ in an i.i.d. fashion[4] as it is the best case scenario for GL as pointed out in Giaretta and Girdzijauskas [7] and such as $\bigcap_{i=1}^{n} \mathscr{D}^i_{\text{training}} = \emptyset$.

ML model. Similarly to Hegedűs et al. [9], each node k trains a (Multinomial) Logistic Regression model to correctly classify images from the dataset described previously. Hegedűs et al. [9] study thoroughly the choice of hyperparameters η (learning rate) and λ (L2 regularization coefficient). In our work, we fix $\eta = 1$ and $\lambda = 0$. We choose those values as they yield best performance on the churn-free Erdős-Rényi case, but, the choice of those 2 values is not trivial and should be studied for all topologies as done by Hegedűs et al [9].

Byzantine attacks. As done by Wu et al. [18] in the FL setting, Byzantine nodes insert a pattern on samples that they also mislabel. In this study, Byzantine nodes insert a 9-pixel trigger pattern to 20% of the image of their dataset, which they relabel to 0. Their goal is to make honest nodes classify marked images as 0, without decreasing the classification performances on untampered data (i.e. without the trigger pattern). As Byzantine nodes attack the GL system by tampering with their dataset to disturb the network, this is called a *data poisoning attack*, and as their goal is to introduce a hidden objective, this kind of attack is also called *backdoor attack*. To assess the performances and the impact of the attack on benign nodes, we draw 2000 images from the test set that we divide into two sets of equal size. The first half will be used to assess the performances on the usual classification task. For the remaining half, we remove all (x,y) where $y = 0$, and apply the same modifications on x that Byzantine nodes do on the remaining samples. We call those two sets *test* and *backdoor* set respectively from now and show samples from both in Figure 1.

In order to choose the Byzantine nodes[5], we will use two strategies: *classical* and *random*. In the *classical* strategy, we select nodes with the highest degree first. In the *random* strategy, we select nodes by randomly sampling without replacement.



(a) Clean instances of the MNIST dataset: The number in the image is the same as the number written above (label)

(b) Example of tampered sample: Notice the label 0 despite the number 0 not written on the image

Fig. 1 Examples of clean and tampered data from the MNIST dataset.

[3] This is done to evaluate the model against unseen data, but close to data that were used for adjusting model's parameters. This allows us to see whether the model generalize well.

[4] This means that data is equally distributed among nodes, every node has approximately 25 images of each number.

[5] We borrow the idea behind these strategies from Magnien et al. [12], where they use these strategies in order to select nodes to be removed from a graph to study the graph connectivity.

Churn. Churn refers to the fact that in a network, devices can join and leave freely. Hegedűs et al [9] used smartphone traces to get a realistic churn scenario (where nodes can disconnect and reconnect). Based on their results, to get close to their churn scenario, nodes have a 20% chance of being online at each round.
Metrics. We are interested here in the average *accuracy* of honest nodes on the test and backdoor sets. *Accuracy* is defined as the number of correct predictions over the total number of predictions, using the dataset as a reference. For the *test* set, the higher, the better: (honest) nodes should perform well on data that has not been marked with the pattern and for the *backdoor* set, it is the opposite: nodes should perform badly on (maliciously) marked images.

4 Simulation results

To implement the study based on the methodology defined in the previous section, we develop an extension of the *gossipy* simulator that makes use of the PyTorch [15] library for its ML part. The *gossipy* simulator enables us to test multiple GL algorithm (PTGLA is already implemented[6]) in a unified manner. However, *gossipy* has two main limitations: it does not take into account the possible existence of Byzantine alongside honest nodes, and it does not allow assessing the performances of both a clean and a corrupted dataset simultaneously. By contrast, our extensions, available (with execution details) on Gitlab[7] address those shortcomings. Simulations were done using Python 3.9.13 or 3.9.15, using CPU only, on Intel Xeon E5-2650v3, and Intel Xeon Gold 6330 machines respectively. We define $n \in \{100, 150\}$ as the number of nodes and f as the number of Byzantine nodes in a simulation (in the worst case, $f = 0.3n$). Each curve shown here is an average over at least 10 random runs. In the following, we present two scenarios, churn and churn-free, starting with the churn-free scenario, i.e. when nodes are always online. Here, Byzantine nodes want honest nodes to classify clean inputs (shown in Section 3) with high accuracy, while also classifying tampered inputs (shown in Section 3) the way Byzantine nodes want, i.e. tampered inputs are classified as 0.
Churn-free scenario. In Figure 2, we represent performances across the 2 sets defined in Section 3 (*test* and *backdoor* sets) when $n = 100$ (resp. $n = 150$) with $f = 30$ (resp. $f = 45$) Byzantine nodes placed with the *random* strategy. We use Byzantine-free simulations as baselines, selecting the number of partitions S that yields the best performances on the test set. On this set (left column of Figure 2), we can see that $S = 1$ and sometimes $S = 4$ (which are the smallest values studied here for S) are special cases across all 4 topologies shown in Figure 2, as honest nodes perform poorly on the usual classification task (compared to the baseline, which is not what Byzantine nodes want), and is very sensitive to the attack as seen in the

[6] https://github.com/makgyver/gossipy/tree/3d655829805fc0dc2f01f5b0862240fca08ffe1c
[7] https://gitlab.lip6.fr/apham/data-poisoning-attacks-in-gossip-learning

right column of Figure 2. This can be explained by the fact that if a Byzantine node directly sends (proactively or reactively) a partition to an honest node, as honest nodes are always online, they always receive and process the partition, and may send it reactively to another node, speeding up the spread of malicious partitions. When Byzantine nodes are placed randomly, if the topology is 20 fan-out, honest nodes are the most sensitive against the attack, while they are more resilient when the topology is Watts-Strogatz.

Besides, on the test set, we observe that when $S \neq 1$ and 4, there are no noticeable differences between the choice of S, as honest nodes perform very closely to the baseline. This can be explained by the i.i.d. data distribution: honest nodes, having the same 'learning material', learn quickly and correctly to classify unaltered inputs. On the backdoor set (right column of Figure 2), still excluding $S = 1$ and 4, we see that the accuracy of honest nodes on the backdoor set is constrained between 0.1 and 0.2, and that, increasing S also increase the resiliency of honest nodes against this particular attack. This can be explained by the fact that parameters that interact with the trigger pattern are too diluted among all partitions as the number of partitions S increase (which is the opposite when S is small), and that Byzantine nodes follow the algorithm (except when training). Hence, corrupted parameters (that interacts with the trigger pattern) are less likely to be updated by honest nodes. However, Byzantine nodes successfully introduce (albeit not as high as they want) the unwanted behavior, without affecting drastically the normal classification task, as we compare the results of the different result of S studied and the baseline on the right column of Figure 2: Byzantine nodes induce from 10 to 20 times more misclassification compared to the baseline. Overall, for the 4 topologies studied, namely Erdős-Rényi, 20 fan-out, 20 random-regular and Watts-Strogatz, when Byzantine nodes are placed randomly, against this particular attack, honest nodes are more resilient when the system is using a (very) high number of partitions S.

In Figure 3, we compare the *classical* and *random* strategy for Byzantine nodes, in systems that have hubs or where the degree distribution is skewed. Usually, for S fixed, the *classical* strategy is more detrimental for honest nodes than the *random strategy*, especially for the Zipf case or when S is low. On the left column of Figure 3, we can see that the accuracy of honest nodes on the test set drop drastically when Byzantines nodes are placed *classically* compared to the *random* strategy. In average, we observe a difference of 6% and 38% on the accuracy for the test set, for Watts-Strogatz and Zipf-based topologies respectively, which is not what Byzantine nodes want for honest nodes, that do classify the way Byzantine nodes want in the backdoor set (right column of Figure 3): in average, we observe a difference of 1% and 48% on the accuracy for the backdoor set between the 2 strategies for Watts-Strogatz and Zipf-based topologies respectively.

Considering the case where Byzantine nodes are placed randomly, Watts-Strogatz, and the other three topologies studied previously, are more suitable for honest nodes compared to the Zipf-based topology as they are more resilient against this attack. We notice that for Watts-Strogatz topology, when S is high, that Byzantine placed classically has almost the same impact as if they were placed randomly, this can be

attributed to the small-world property of the topology and the fact that the backdoor is too diluted.

Interestingly, for Zipf-based topology, when Byzantine nodes are placed randomly, excluding $S = 1$, on the right column, it sometimes seems detrimental for honest nodes that the system use a high number of partitions S compared to the other 4 topologies already studied previously.

Churn scenario. Here, we study the case where nodes can disconnect and reconnect as described in Section 3 and have a 20% chance to be online.

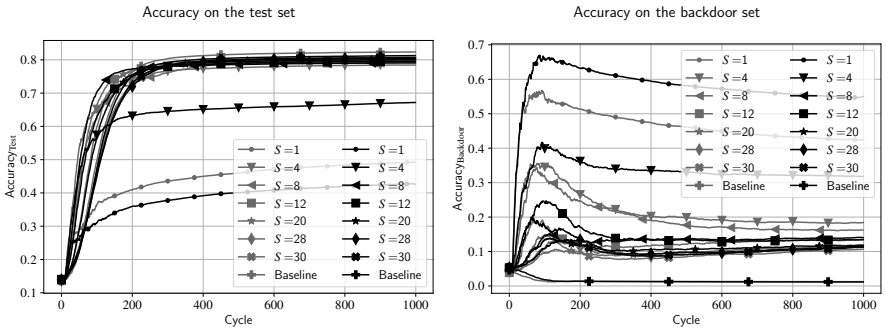
In Figure 4, we fix the number of partitions to $S = 8$ and number of nodes to $n = 150$, and study the effect of Byzantine placement strategy and numbers up to $f = 45$ when nodes degree follow a Zipf law distribution. We can see, on the right column of Figure 4, that Byzantine nodes, when placed with the *classical* and *random* with $f = 5$, for the random and classical strategy, it is almost as if there is no attack (case $f = 0$). While the *classical* strategy is more harmful to the network for f fixed, considering the backdoor set, we note the fact that $f = 40$ Byzantine nodes selected with the *random* strategy is about 5% more harmful than $f = 20$ Byzantine nodes selected with the *classical* strategy.

In Figure 5, we again study $n = 100$ (resp. $n = 150$) and $f = 30$ (resp. $f = 45$), with the *random strategy*, similarly to Figure 2. We see that honest nodes perform as well as in the *churn-free* scenario on the test set (left column of Figure 5), still due to the fact that data is distributed in an i.i.d. fashion. In this scenario, honest nodes are also slower compared to the churn-free scenario, which is also expected as they are offline most of the time, hence, they are less learning exchanges between nodes for a given number of rounds compared to the churn-free scenario.

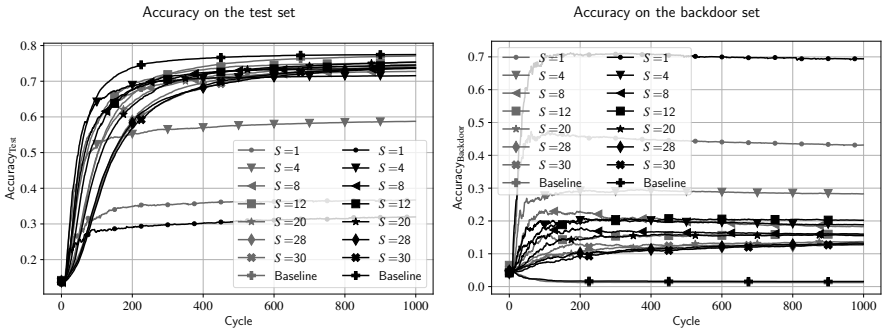
Interestingly, while $S = 1$ is again a special case here, unlike the churn-free scenario, honest nodes classify badly clean inputs at the beginning, but they converge to a much better solution at the end, as they successfully manage to classify clean inputs with a better success rate and are less affected by the attack compared to the churn-free scenario. This is due to the fact that all nodes are most of the time offline, Byzantine nodes included, hence, the attack is not as powerful as in the churn-free scenario, but we notice, that the results are close with the other value of S studied. In the journal version of this work, we are planning to study the churn scenario with the assumption that Byzantines nodes are always online. Overall, we notice that values presented in the churn-free scenario (Figure 2) and the churn scenario (Figure 5) are relatively close. This might be explained by the fact that the churn is probabilistic, combining with the fact that the choice of the i^{th} to send is random, hence, over time, the churn-free scenario, looks very similar to the churn-free case (except when $S = 1$).

5 Conclusion

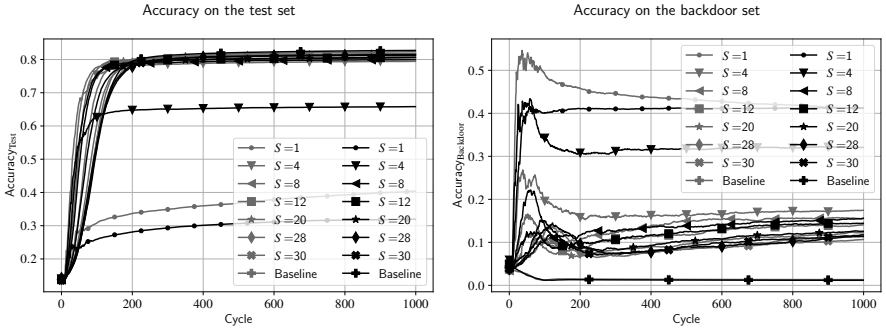
In this paper, we proposed a methodology to study the resilience of gossip learning in the presence of poisoning attacks. As case study we target the compression mechanism of GL algorithm proposed by Hegedűs et al. [9]. We investigate its resilience



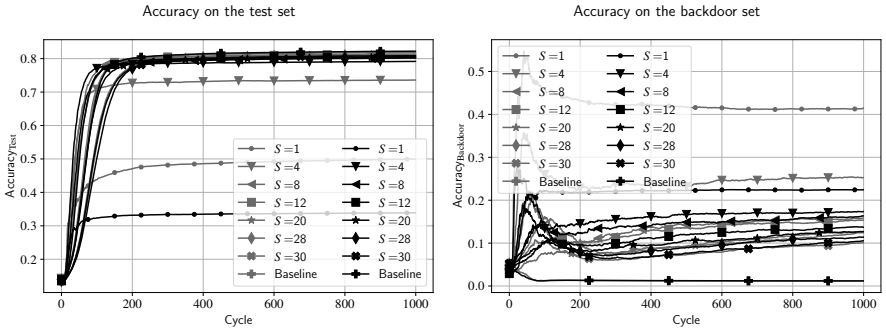
(a) Erdős-Rényi



(b) 20 fan-out

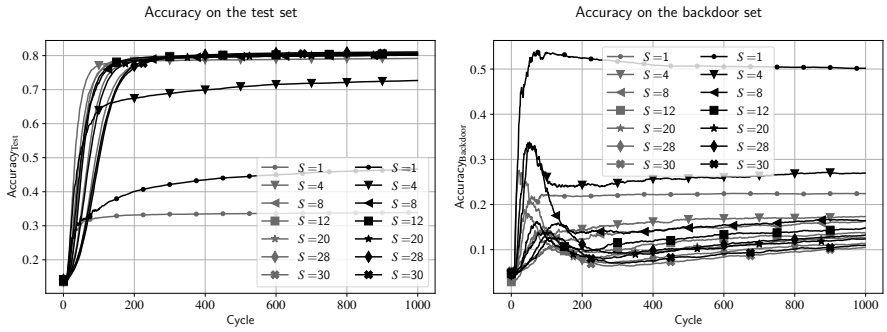


(c) random 20-regular

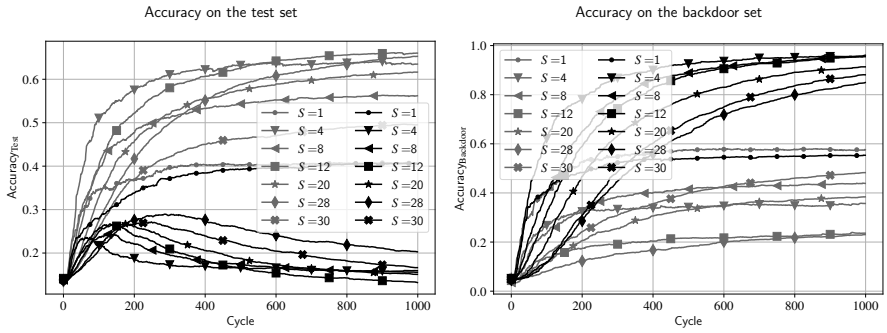


(d) Watts-Strogatz

Fig. 2 Accuracy on the test set (left, higher is better) and backdoor set (right, lower is better) for different topologies with $n = 100$ (gray) and $n = 150$ (black) with random Byzantine placement strategy in a churn-free system with $f = 30$ and 45 respectively. 'Baseline' curves represent the results in Byzantine-free simulations, with the best choice of S among values studied here with Byzantine.



(a) Watts-Strogatz



(b) Zipf law based degree distribution

Fig. 3 Accuracy on the test set and backdoor set for different topologies with $n = 150$ and $f = 45$ with random Byzantine placement strategy (gray) and classical Byzantine placement strategy (

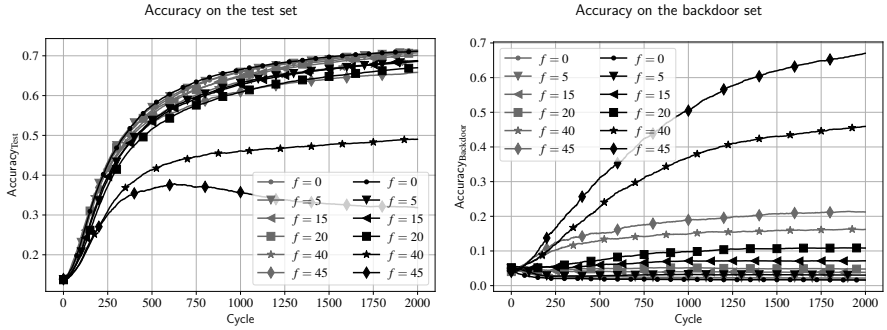
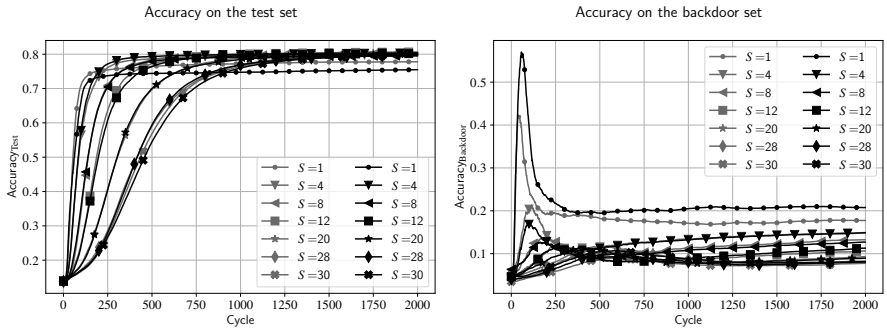
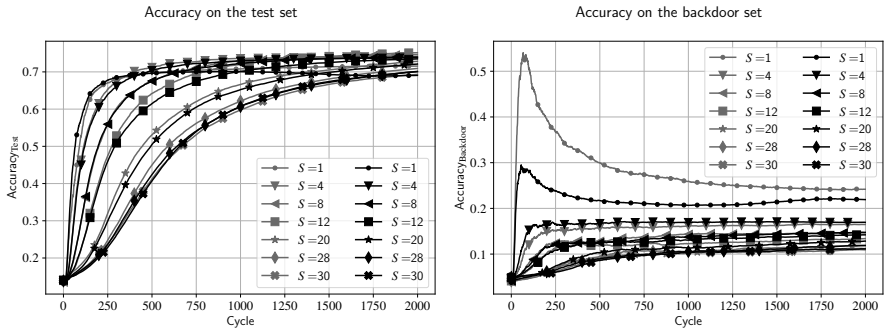


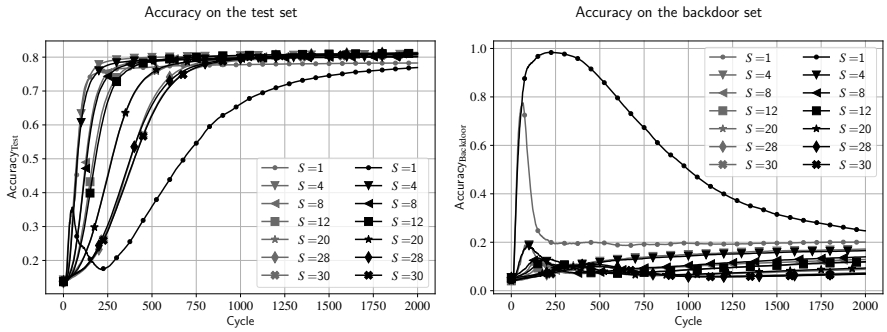
Fig. 4 Accuracy on the test and backdoor set for $n = 150$, $S = 8$ for $f \in \{0,5,15,20,25,40,45\}$ with the random (gray) and classical (black) placement strategy when nodes degree follow a Zipf law.



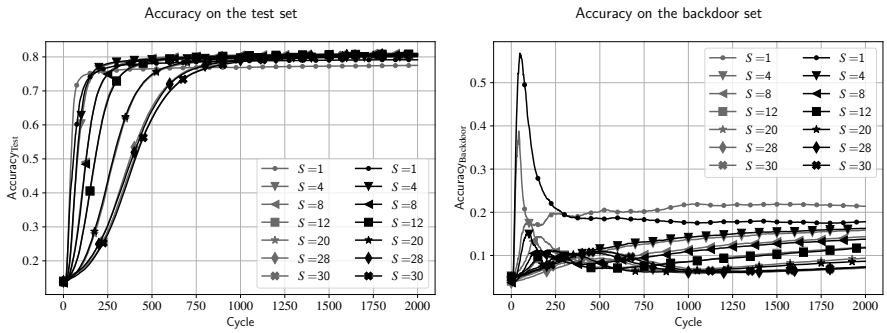
(a) Erdős-Rényi



(b) 20 fan-out



(c) random 20-regular



(d) Watts-Strogatz

Fig. 5 Accuracy on the test set and backdoor set for different topologies with $n = 100$ (gray) and 150 (black) with $f = 30$ and 45 respecively.

in a broad range of topologies (e.g. Erdős-Rényi, 20 fan-out, random 20-regular, Watts-Strogatz and Zipf-based graph) in both churn-free and churn scenarios. Our findings show that the communication optimizations and the choice of the underlying topology are not always favorable to the honest nodes. Moreover, the distribution of the Byzantine nodes has also a strong impact on the accuracy metric.

Usually, in the churn-free and churn cases, when Byzantine nodes are placed randomly, the use of a low number of partitions (i.e. bigger messages), is usually detrimental for honest nodes. This is not necessarily true if the topology is Zipf-based, in the churn-free scenario.

In the churn-free scenario, when Byzantine nodes are placed using the *classical* strategy in Watts-Strogatz and Zipf topologies, the attacks can exhibit a completely different impact on the network compared to when they are placed randomly. We observe that in average, Byzantine nodes cause a drop of 1% and 38% on the usual classification task and a 6% and 48% increase on the backdoor task respectively.

For honest nodes, when Byzantine nodes are placed randomly, we do not observe noticeable differences between the churn-free and churn scenario (except when nodes use bigger messages, as they are more resilient against the attack in the latter scenario), this might be attributed to the probabilistic churn and probabilistic nature of the GL algorithm.

In the future we plan to extend this work to more complex datasets and data distribution, and study other proposals for GL algorithm including some that take into account the possibility of poisoning data and models.

References

[1] Y. Belal, A. Bellet, S. B. Mokhtar, and V. Nitu. PEPPER: Empowering User-Centric Recommender Systems over Gossip Learning. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 6(3):101:1–101:27, Sept. 2022.

[2] E. T. M. Beltrán, M. Q. Pérez, P. M. S. Sánchez, S. L. Bernal, G. Bovet, M. G. Pérez, G. M. Pérez, and A. H. Celdrán. Decentralized Federated Learning: Fundamentals, State of the Art, Frameworks, Trends, and Challenges. *IEEE Communications Surveys & Tutorials*, pages 1–1, 2023.

[3] J. Bernstein, Y.-X. Wang, K. Azizzadenesheli, and A. Anandkumar. signSGD: Compressed Optimisation for Non-Convex Problems. In *Proceedings of the 35th International Conference on Machine Learning*, pages 560–569. PMLR, July 2018.

[4] J. Bernstein, J. Zhao, K. Azizzadenesheli, and A. Anandkumar. signSGD with Majority Vote is Communication Efficient and Fault Tolerant. In *International Conference on Learning Representations*, Sept. 2018.

[5] G. Danner, I. Hegedűs, and M. Jelasity. Improving Gossip Learning via Limited Model Merging. In *Advances in Computational Collective Intelli-*

gence, Communications in Computer and Information Science, pages 351–363, Cham, 2023. Springer Nature Switzerland.

[6] P. Erdős and A. Rényi. On random graphs I. *Publicationes Mathematicae Debrecen*, 6:290–297, 1959.

[7] L. Giaretta and Š. Girdzijauskas. Gossip Learning: Off the Beaten Path. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 1117–1124, Dec. 2019.

[8] J. Guo, Y. Zuo, C.-K. Wen, and S. Jin. User-Centric Online Gossip Training for Autoencoder-Based CSI Feedback. *IEEE Journal of Selected Topics in Signal Processing*, 16(3):559–572, Apr. 2022.

[9] I. Hegedűs, G. Danner, and M. Jelasity. Decentralized learning works: An empirical comparison of gossip learning and federated learning. *Journal of Parallel and Distributed Computing*, 148:109–124, Feb. 2021.

[10] Y. LeCun, C. Cortes, and CJ. Burges. MNIST handwritten digit database. *ATT Labs [Online]*, 2, 2010.

[11] P. Liu, X. Xu, and W. Wang. Threats, attacks and defenses to federated learning: Issues, taxonomy and perspectives. *Cybersecurity*, 5(1):1–19, Dec. 2022.

[12] C. Magnien, M. Latapy, and J.-L. Guillaume. Impact of random failures and attacks on Poisson and power-law random networks. *ACM Computing Surveys*, 43(3):13:1–13:31, Apr. 2011.

[13] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas. Communication-Efficient Learning of Deep Networks from Decentralized Data, Feb. 2016.

[14] R. Ormándi, I. Hegedűs, and M. Jelasity. Gossip learning with linear models on fully distributed data. *Concurrency and Computation: Practice and Experience*, 25(4):556–571, 2013.

[15] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[16] Y. Qu, C. Xu, L. Gao, Y. Xiang, and S. Yu. FL-SEC: Privacy-Preserving Decentralized Federated Learning Using SignSGD for the Internet of Artificially Intelligent Things. *IEEE Internet of Things Magazine*, 5(1):85–90, Mar. 2022.

[17] D. J. Watts and S. H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393(6684):440–442, June 1998.

[18] C. Wu, X. Yang, S. Zhu, and P. Mitra. Mitigating Backdoor Attacks in Federated Learning, Jan. 2021.

[19] G. Xia, J. Chen, C. Yu, and J. Ma. Poisoning Attacks in Federated Learning: A Survey. *IEEE Access*, 11:10708–10722, 2023.