Topology-Driven Defense: Detecting Model Poisoning in Federated Learning with Persistence Diagrams

Narges Alipourjeddi and Ali Miri,
Department of Computer Science, Toronto Metropolitan University, Toronto, Canada
narges.alipourjeddi, ali.miri@torontomu.ca

Keywords: Federated Learning, Model Poisoning Attacks, Model Aggregation,Topological Data Analysis, Persistence homology, Anomaly detection ,Security Vulnerabilities, Robustness in FL, Machine Learning Security, Model Integrity

Abstract: Federated Learning (FL) has emerged as a transformative approach for training machine learning models across decentralized data sources while keeping client data localized. Despite its advantages, FL systems remain vulnerable to various attacks and anomalies, including model poisoning attacks, which compromise the integrity of the global model. In this paper, we introduce a novel approach for detecting such attacks by leveraging persistence diagrams derived from topological data analysis (TDA). Our method provides a comprehensive solution for identifying anomalies in the training process by computing persistence diagrams in high-dimensional spaces, effectively addressing the challenges of analyzing complex neural network architectures. Through extensive experiments, we demonstrate that our approach achieves high accuracy in detecting and mitigating attacks, even under non-IID and highly unbalanced data distribution scenarios. We evaluate our method across various datasets and attack scenarios, and the results validate its robustness and effectiveness, establishing it as a promising solution for enhancing the security of federated learning environments.

1 INTRODUCTION

Recent advances in machine learning have traditionally relied on large, centralized datasets (Chowdhery et al., 2023). However, this approach faces significant challenges, particularly when dealing with sensitive or private information that must remain distributed across various clients, such as mobile devices, healthcare institutions, or smart buildings (Bhat et al., 2023). To address these challenges, Federated Learning (FL) has emerged as a promising paradigm for leveraging distributed data while maintaining data locality (McMahan et al., 2017). FL allows multiple parties to collaboratively train machine learning models without sharing raw data, instead exchanging only model updates. By enabling local model training on edge devices using local datasets and aggregating only model parameters, FL offers a potential solution to privacy concerns in data-sensitive applications. However, recent research has revealed that FL systems are vulnerable to various attacks, including security and privacy attacks.

Once the FL is attacked, it can lead to privacy leakage, model damage, system robustness degradation, and other adverse effects, ultimately causing a loss of user trust.

To address these vulnerabilities, numerous methods have been proposed to defend against such attacks and address the security and privacy anomalies in FL (Ma et al., 2022). For example, in the context of security attacks, the (Xu et al., 2024) checked for client updates in each iteration and discarded potentially malicious clients as the server aggregated updates. In terms of privacy attacks, (Gao et al., 2021) proposed a defense against reconfiguration by searching for privacy-preserving transformation functions and pre processing training samples with such functions to ensure an excellent performance from the trained model.

In FL settings, sensitive information is vulnerable to attacks on clients, communication processes, and the central server. At the client level, attackers may inject malicious updates or extract sensitive data through methods like data poisoning or backdoor attacks. Data poisoning involves sending incorrect model parameters to the server, leading to compromised global model performance. Backdoor attacks, more subtle, embed malicious functionality in the global model while maintaining its performance on main tasks, triggering undesirable behaviors only with specific inputs.

During communication, adversaries may intercept or tamper with data transmitted between clients and the server, causing data leakage or integrity breaches.

The central server in FL can be directly targeted to compromise the integrity, confidentiality, and availability of the process. Unlike client-side attacks, these server-level attacks, such as model poisoning, manipulate aggregation by adding noise, altering weights, or introducing patterns that degrade global model performance. This paper proposes a novel method for detecting central server attacks using advanced anomaly detection techniques, including topological data analysis (TDA). By focusing on server-side detection, our approach aims to improve FL security and reliability while preserving its inherent privacy benefits.

TDA and persistence diagrams offer a promising approach for detecting anomalies in model updates. By capturing the topological features of data, persistence diagrams serve as a robust tool for identifying irregularities in the training process.(Carrière et al., 2020). By analyzing the persistence diagrams of global model updates, it is possible to detect and mitigate attacks in FL environments. This method leverages the unique properties of TDA to identify deviations from normal model behaviour, enabling timely detection of malicious activities. Our research demonstrates the effectiveness of this approach in enhancing the security and robustness of FL systems. As the field evolves, researchers are exploring novel techniques to improve attack detection in FL, balancing the trade-offs between model performance, communication efficiency, and security. These advancements aim to make FL a more robust and secure framework for collaborative machine learning across diverse and sensitive datasets.

Existing models are typically trained on known data types, which are assumed to be Independent and Identically Distributed (IID). However, in real-world scenarios, data often exhibits significant heterogeneity, and models cannot anticipate the complete spectrum of data distribution. Data frequently manifests in Non-Independent and Identically Distributed (Non-IID) forms. TDA offers powerful tools for understanding the structure of data. Integrating TDA into FL can provide insights into data distributions, enhance model performance, and address issues like data heterogeneity.

In this paper, we present a novel approach for detecting attacks in FL systems by leveraging persistence diagrams to capture and analyze topological features of model updates. Our method integrates topological data analysis to extract persistent homology features and compare differences. Section 2 covers the background and notations, Section 3 reviews related work, Section 4 presents our framework, Section 5 demonstrates its practical capabilities, and Section 6 concludes with a summary and insights.

2 PRELIMINARIES

Federated Learning Framework

Federated Learning (FL) is a decentralized machine learning approach designed to preserve data privacy by enabling model training across multiple clients (e.g., mobile devices or organizations) without transferring local data to a central server (McMahan et al., 2017). Each client trains a model locally and shares updates, such as gradients or weights, with a central server, which aggregates them to create a global model.

Formally, the goal of FL is to obtain a global objective function that aggregates local objectives across N clients:

$$min_w F(w) = \sum_{k=1}^{N} p_k F_k(w)$$

where w is the global model's parameter vector and $F_k(w)$ is a local loss function for client k.

$$F_k(w) = \frac{1}{n_k} \sum_{i \in D_k} l(w, x_i, y_i)$$

with $n_k = |D_k|$ being the number of data samples for client k and $l(w, x_i, y_i)$ being a loss for data data sample (x_i, y_i) and $p_k = \frac{n_k}{\sum_{j=1}^{N} n_j}$ represents the relative weight of client k's data in the global objective. Each client k updates its local model based on the current global model w^t received from the server. Using Stochastic Gradient Descent (SGD) (Xiong et al., 2021), this can be expressed as:

$$w_k^{t+1} = w_k^t - \eta \nabla F_k(w_k^t)$$

where w_k^t is the local model of client k at iteration t, η is a global learning rate applied by the parameter server, to control learning speed and $\nabla F_k(w_k^t)$ is the gradient of the local loss function with respect to w_k^t.

The central server aggregates the local models from the participating clients to update the global model. This is commonly done using Federated Averaging (FedAvg)(McMahan et al., 2017):

$$w^{t+1} = \sum_{k=1}^{N} p_k w_k^{t+1}$$

where w_k^{t+1} is the updated model from client k after local training.

Central Server Attacks in FL

In a FL setup, the central server is vital for aggregating updates from clients and distributing the global model. However, this centralization also makes the server vulnerable to attacks. Key threats include malicious server activities and non-robust aggregation processes. The server must be rigorously secured to prevent exploitation of potential vulnerabilities.

Malicious server attacks occur when adversaries manipulate the aggregation process or global model to degrade performance or introduce vulnerabilities. Noise injection is a prominent attack method, where attackers add crafted noise to disrupt model performance. This can include:

- Random Noise Injection: Using Gaussian-distributed noise $\varepsilon \sim N(\mu, \sigma^2)$ to destabilize the aggregation process.
- Strategic Noise Injection: Introducing tailored noise to subtly influence the global model, potentially causing specific biases or performance degradation.

Noise injection attacks are particularly insidious because they can be subtle and hard to distinguish from natural variations in the FL process. They exploit the trust placed in the central server and can have long-lasting effects on the model's performance and reliability. Defending against such attacks requires a multifaceted approach. By using TDA methods, we can enhance the robustness of FL systems.

Persistence Diagram

Persistence Diagram summarizing topological features within a dataset across varying scales. They capture the evolution of features such as connected components (0-dimensional), loops (1-dimensional), and voids (2-dimensional) as the scale of analysis changes. This is achieved through a filtration, a sequence of nested topological spaces (e.g., simplicial complexes) that progressively incorporate more data. This process reveals how topological features emerge, evolve, and eventually disappear within the filtration. For instance, the Vietoris-Rips complex, a common simplicial complex, is constructed based on a distance threshold d among points in a metric space. A Vietoris-Rips filtration tracks how the topological structure evolves as d increases. Another crucial concept is homology, a mathematical framework for identifying and counting "holes" in spaces. Homology classifies features by their dimensionality: 0-dimensional for connected components, 1-dimensional for loops, and so on. The technique of persistent homology tracks these features across the filtration, recording their birth (appearance) and death (appearance).

Persistence diagrams visualize these data by plotting each feature as a point, with its birth on the x-axis and death on the y-axis. This visualization effectively highlights the robust topological features within the dataset, distinguishing them from transient ones.

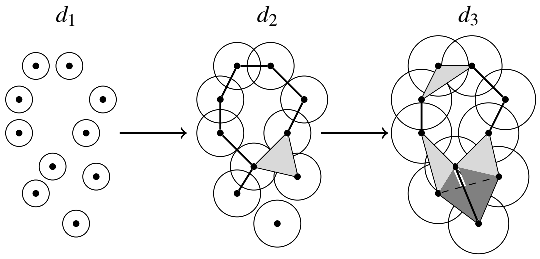

Figure 1: The three step filtration of Vietoris-Rips complex on the set of 10 points with increasing radius.

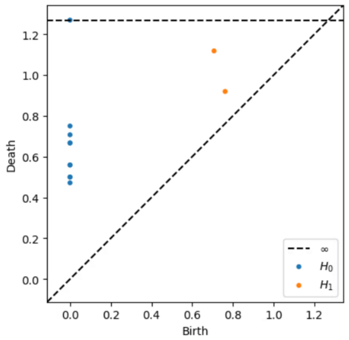

Figure 2: The persistence diagram corresponds to the filtration shown in the Figure 1. Blue points represent connected components, while orange points represent loops.

Figure 1 shows the three steps of Vietoris-Rips filtration in the set of 10 data points and Figure 2 shows the persistence diagram of our dataset. We detect potential attacks by analyzing key features in persistence diagrams and their topological signatures. Using the bottleneck distance, which measures the largest differences in feature birth and death times between diagrams, we detect deviations caused by attacks, distinguishing normal from anomalous.

3 RELATED WORKS

In FL, the central server's aggregation algorithm is vital for enabling collaborative learning without data sharing. However, adversaries can exploit vulnerabilities in these methods, particularly through model poisoning attacks. This section explores significant related works addressing such threats.

Several model poisoning attack strategies have been proposed to compromise FL systems. *Bagdasaryan et al.*(Bagdasaryan et al., 2020) introduced a model replacement attack to scale malicious updates before submission. *Xie et al.*(Xie et al., 2024) proposed a distributed backdoor attack (DBA), where adversaries embed local triggers in separate clients. *Fang et al.* (Fang et al., 2020) demonstrated the effectiveness of local model poisoning, where a few compromised clients can significantly affect the global model. *Wang et al.* (Wang et al., 2020) analyzed aggregation methods, highlighting their susceptibility to Byzantine attacks and the need for robust aggregation techniques. Model poisoning attacks can have severe consequences on FL systems. *kairouz et al.* (Kairouz et al., 2021) showed that targeted attacks could introduce specific biases or backdoors into the aggregated model, affecting predictions. Defending against these attacks poses significant challenges. While significant progress has been made in understanding and addressing model poisoning attacks in FL, developing comprehensive and effective strategies remains an active area of research. Recent advancements in computational topology, including persistent homology (PH), have been applied to machine learning. It was leveraged to analyze the topological feature of various kinds of data structures. In the (Alipourjeddi and Miri, 2023), proposed a method to compare generative models by persistent diagrams for analyzing the similarities and differences in the birth and death times of topological features, to determine how each model captures the underlying structure of the data at various scales. *Alipourjeddi and Miri* (Alipourjeddi and Miri, 2024) experimented on high dimensional datasets to unveil hidden patterns and captures the persistent topological features of the data, allowing to study its shape and structure across different scale. Ma and Gao (Ma and Gao, 2024) trained a classifier based on the PH features of neural network models and composing a secure FL mechanism. Building on previous research, this paper proposes a PD-based method for detecting attacks in the uploads from

4 METHODOLOGY

Our methodology for investigating model poisoning attacks on the central server in FL using persistence diagrams includes the following phases:

- FL Setup: Establish an FL environmen using a suitable framework, such as TensorFlow Federated. This involves defining the model architecture, determining the number of participating clients, and specifying the communication rounds. Appropriate datasets, such as MNIST and CIFAR-10, are carefully selected under a non-IID setting to ensure relevance and comparability with existing research.
- Attack Design: Develop targeted model poisoning attacks to manipulate updates at the server, ensuring persistence across training rounds while minimizing detectability.
- TDA Application: A key innovation in our methodology is applying TDA to generate persistence diagrams from model updates, effectively capturing the topological features of both benign and poisoned updates. This step is foundational for subsequent analysis and detection efforts.
- Detection Mechanism: To detect attacks, we compute persistence diagrams for the aggregated weights after each round. If the bottleneck distance between consecutive diagrams exceeds a threshold, it is flagged as a potential attack.
- Performance Evaluation: The performance of the persistence diagram-based detection method is evaluated using metrics, including detection rate and false positive rate. Experiments are conducted with varying noise levels, communication rounds, client participation, and non-IID data distributions to assess the robustness of the approach.

Through this comprehensive methodology, we aim to provide a detailed investigation of model poisoning attacks at the central server in federated learning while demonstrating the potential of persistence diagrams as a novel tool for enhancing the security of such systems.

5 EXPERIMENTAL EVALUATION

In this part, we carry out extensive experiments to demonstrate the performance of our mechanism. We have implemented the federated learning environment with FedAvg introduced in (Sun et al., 2022) which allows multiple edge devices to collaboratively train a model without sharing data. It involves running stochastic gradient descent (SGD) locally on client devices and then averaging the model updates on a central server. The experiment modifies a model poisoning attack by adding noise to the aggregate weight, which is a technique used to compromise the integrity of the federated learning process.The Persistent Diagram (PD) is calculated using the Ripser package, which is known for its efficiency in computing persistent homology introduced by (Bauer, 2021). The bottleneck distance is used to measure the difference

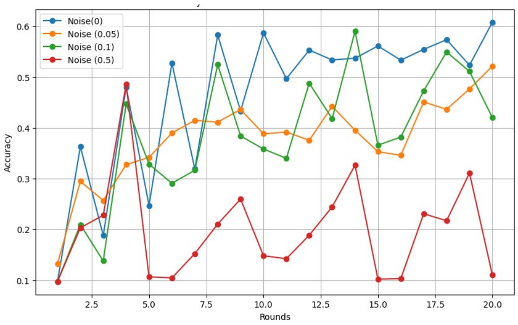

Figure 3: Accuracy vs. Rounds: Performance Evaluation on the MNIST Dataset

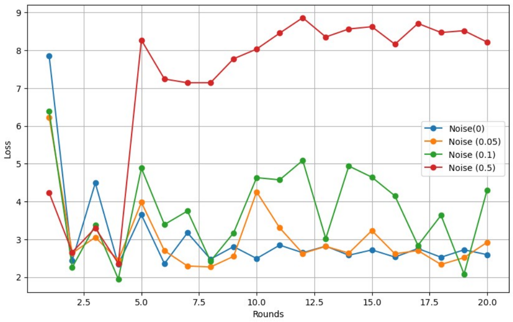

Figure 4: Loss vs. Rounds: Performance Evaluation on the MNIST Dataset

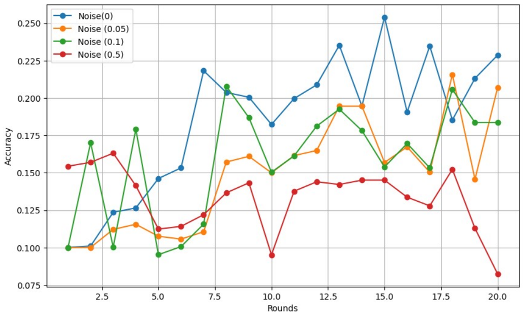

Figure 5: Accuracy vs. Rounds: Performance Evaluation on the CIFAR-10 Dataset

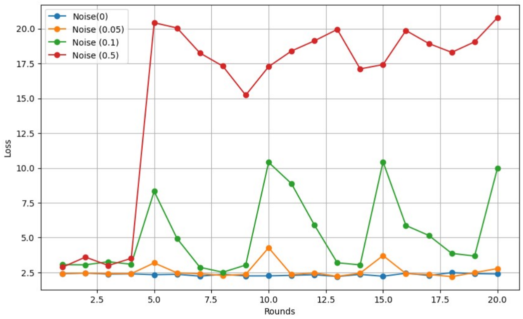

Figure 6: Loss vs. Rounds: Performance Evaluation on the CIFAR-10 Dataset

between PDs, helping to identify the presence and impact of attacks on the federated learning model (Che et al., 2024). This setup aims to demonstrate how adding noise to aggregated weights can affect federated learning models and uses advanced mathematical tools to quantify these effects.

Datasets: In this section, we begin by generating a synthetic dataset with 1,000 points to simulate clients in a non-IID federated learning (FL) setting. Subsequently, we perform extensive experiments using real-world datasets, MNIST and CIFAR-10, to validate our analysis of privacy leakage in FL. Both MNIST and CIFAR-10 are widely-used datasets for classification tasks, each containing 10 classes. To emulate a non-IID environment in the FL system, we distribute the datasets among clients based on class labels. This ensures that the local datasets assigned to each client follow heterogeneous distributions, effectively reflecting real-world non-IID scenarios.

Attack setting: We explore server-side attacks in FL, where an adversary manipulates the global model aggregation process by injecting additional noise. This deliberate interference introduces malicious updates, degrading the system's performance before the manipulated global model is broadcast back to the clients. Specifically, we simulate a global model poisoning attack, characterized by the following noise injection details: Adjust the noise based on the number of clients and the total number of iteration, considering the initialized model (Xiong et al., 2021). In our experiment, we simulate an attack scenario over 20 rounds, with 10 clients participating in each round for all three datasets. We introduce adversarial behavior every fifth round by injecting noise into the aggregation process. These attacks aim to degrade the model's utility by either biasing the global model or disrupting convergence. To assess the model's resilience, we varied the intensity of the attacks using two parameters: the proportion of clients and the magnitude of the adversarial updates. Our strategy was to evaluate how much adversarial influence the model could tolerate while maintaining acceptable utility. We focused on identifying the thresholds at which the model began to degrade significantly under attack. Convergence slowed down as the attack intensity increased, especially during rounds where malicious updates were introduced. However, the training loss continued to decrease after each round of federated training, indicating that the model was still converging, albeit more slowly. Figures 3 and 4 demonstrate that for the MNIST dataset with 10 expected clients per round, the model tolerated a noise multiplier of up to 0.05 without significant degradation in model quality. A noise multiplier of 0.1 introduced slight degradation, while a multiplier of 0.5 caused

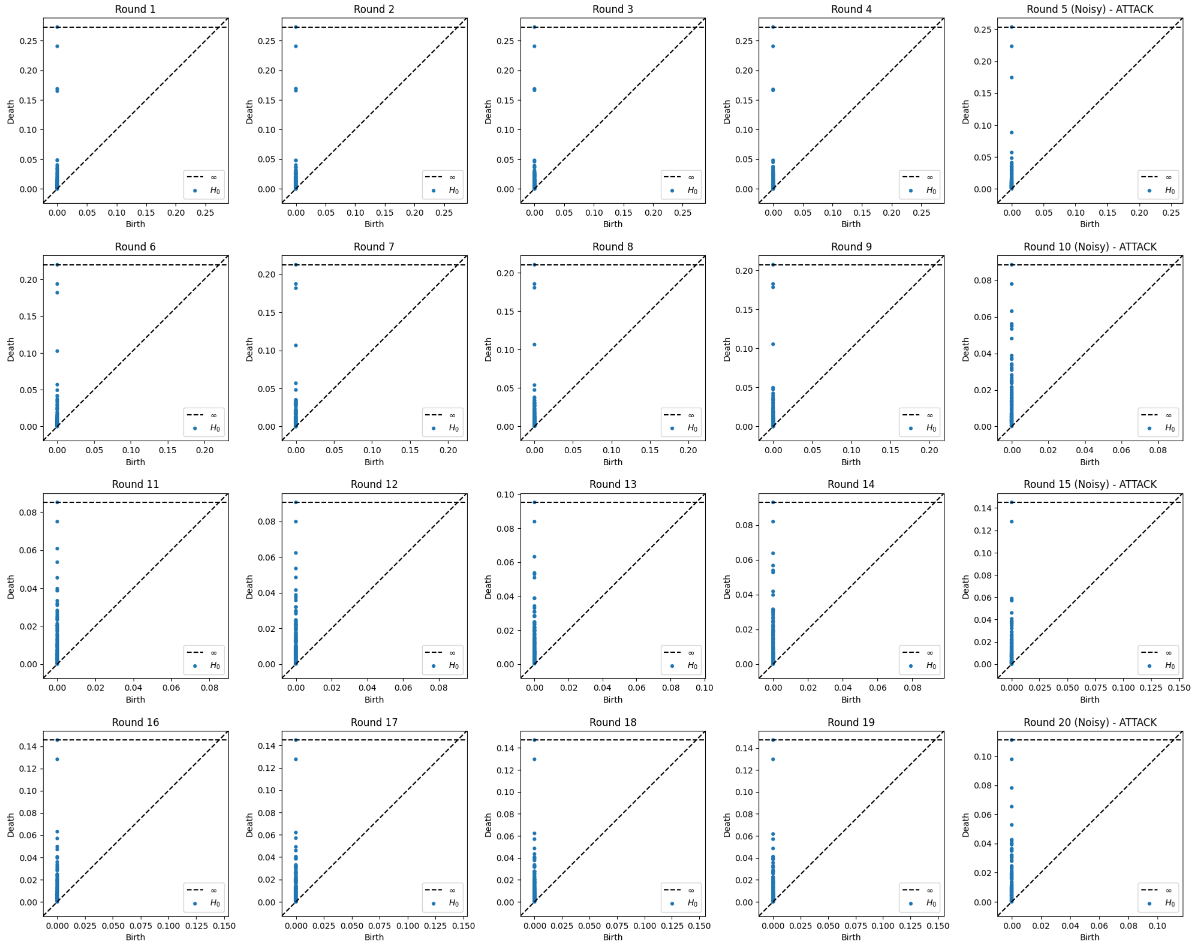

Figure 7: Persistence diagrams for the first 20 rounds on the MNIST dataset. Noise was introduced in rounds 5, 10, 15 and 20 leading to detected attacks with threshold 0.05 revealing significant changes in the topological features of the aggregated model weights.

the model to diverge completely.

Similarly, for the CIFAR-10 dataset, we applied the same approach to identify tolerable noise levels within our attack setting. Unlike MNIST, CIFAR-10 required a more sophisticated model architecture due to its higher complexity, including color channels and intricate image features. To achieve better accuracy, we developed a model with additional convolutional layers featuring larger filter sizes, along with advanced components such as batch normalization and dropout layers. These enhancements were designed to address the greater variability and richness of the CIFAR-10 dataset, ensuring improved performance while maintaining robustness under noise.

Figures 5 and 6 demonstrate that, for the CIFAR-10 dataset, the model remains robust with a noise multiplier of up to 0.05, showing no significant degradation in performance. However, increasing the noise multiplier to 0.1 introduces moderate performance degradation, and a multiplier of 0.5 causes the model to diverge entirely.

Attack detection with PDs: To detect attacks in our system, we use persistence diagrams, a tool that captures the shape and structure of the data. Our approach leverages the sensitivity of persistence diagrams to perturbations in the aggregated model weights, allowing us to identify potential attacks. We construct persistence diagrams from the aggregated weights of the federated model after each round of training. The weight matrices are flattened into a high-dimensional point cloud, and we compute the persistent homology using the Vietoris-Rips filtration. This process captures the topological features of the weight distribution across multiple scales.

Our attack detection mechanism is based on the observation that malicious perturbations in the aggregated weights alter the topological structure of the persistence diagram. Key aspects include:

Dataset	Rounds	Noise Level							
		0.0		0.05		0.1		0.5	
		Detection Rate	False Positive Rate	Detection Rate	False Positive Rate	Detection Rate	False Positive Rate	Detection Rate	False Positive Rate
Synthetic Data	20	0.00	0.00	0.75	0.0625	1.00	0.0625	1.00	0.0625
MNIST		0.60	0.02	0.75	0.05	1.0	0.0	0.95	0.15
CIFAR-10		0.55	0.10	0.65	0.15	0.75	0.18	0.85	0.22
Synthetic Data	50	0.00	0.00	0.8	0.05	0.6	0.0	1.00	0.1
MNIST		0.70	0.01	0.80	0.05	0.85	0.1	0.95	0.05
CIFAR-10		0.80	0.05	0.85	0.1	0.	0.00	0.00	0.00
Synthetic Data	100	0.00	0.0125	0.65	0.0	0.75	0.00	1.00	0.0875
MNIST		0.75	0.00	0.80	0.05	0.90	0.05	1.0	0.0675
CIFAR-10		0.60	0.08	0.70	0.12	0.85	0.14	0.95	0.15

Table 1: Performance metrics for detecting attacks using persistence diagrams under varying noise levels across three datasets.

- Bottleneck Distance: We compute the bottleneck distance between the persistence diagrams of consecutive training rounds.
- Attack Threshold: We establish a threshold to flag ongoing attacks when the bottleneck distance exceeds it.

This methodology allows us to effectively detect attacks by analyzing topological variations in the model's aggregated weights.

Figure 7 shows persistence diagrams for the first 20 rounds of federated learning in the MNIST dataset. Each subplot represents a round, with the x-axis showing the birth times and the y-axis showing the death times of topological features. In particular, in rounds 5, 10, 15 and 20, where noise level of 0.1 was introduced to the aggregated weights at the server, we observe significant changes in the distribution of points. This pattern aligns with our attack detection threshold of 0.05, suggesting that the injected noise has measurably altered the topological structure of the model weights. The progression of diagrams across rounds illustrates the evolution of the model's topological features throughout the federated learning process, with rounds 5, 10, 15 and 20 clearly demonstrating the impact of simulated attacks on the model's structure.

Quantitatively, Table 1 summarizes the performance metrics for detecting attacks using persistence diagrams under varying noise levels across three datasets: Synthetic Data, MNIST, and CIFAR-10. For synthetic data, the detection rate increases significantly with noise, reaching perfect detection (1.00) at higher noise levels (0.1 and 0.5) across all round configurations, while maintaining a relatively low false positive rate (0.0625-0.1). The MNIST dataset shows robust performance, with detection rates ranging from 0.60 to 1.0 and generally low false positive rates, particularly at 100 rounds where it achieves perfect detection at 0.5 noise level with a 0.0675 false positive rate. CIFAR-10 presents more challenges, with lower detection rates at low noise levels but improving performance as noise increases, reaching up to 0.95 detection rate at 0.5 noise level in 100 rounds, albeit with higher false positive rates compared to other datasets. Interestingly, CIFAR-10 shows anomalous behavior at 50 rounds with 0.1 and 0.5 noise levels, suggesting potential instability or data peculiarities. Overall, the method's performance tends to improve with increased noise levels and more federated learning rounds, demonstrating its effectiveness in detecting attacks in federated learning systems, especially under higher noise conditions.

6 CONCLUSION

In this paper, we introduced a novel approach to detect server-side attacks in FL using persistence diagrams. Our method successfully identified noise injections every fifth round with a threshold of 0.05, demonstrating its effectiveness in capturing topological changes in aggregated model weights. Our work contributes to the growing body of research on federated learning security. While existing methods focus on client-side attacks or network intrusions, our approach addresses the critical issue of server-side manipulations. The use of topological data analysis offers a unique perspective on model integrity, complementing traditional distance-based or gradient-based anomaly detection techniques.

The results show that persistence diagrams can effectively visualize and quantify the impact of adversarial perturbations on model structure. This provides researchers and practitioners with a new tool for monitoring federated learning systems and detecting subtle attacks that may evade other detection methods.

Further research is needed to explore the scalability of our approach with more complex models and investigate higher-dimensional homology to capture intricate topological features in model updates. Additionally, evaluating the method's performance against more sophisticated attack strategies would help assess its robustness in real-world scenarios.

7 ACKNOWLEDGEMENTS

This work was partially funded by the Canada First Research Excellence Fund (CFREF) Bridging Divides program.

REFERENCES

Alipourjeddi, N. and Miri, A. (2023). Evaluating generative adversarial networks: A topological approach. In *2023 International Conference on Computing, Networking and Communications (ICNC)*, pages 202–206. IEEE.

Alipourjeddi, N. and Miri, A. (2024). Preserving privacy in high-dimensional data publishing. In *ICISSP*, pages 845–852.

Bagdasaryan, E., Veit, A., Hua, Y., Estrin, D., and Shmatikov, V. (2020). How to backdoor federated learning. In *International conference on artificial intelligence and statistics*, pages 2938–2948. PMLR.

Bauer, U. (2021). Ripser: efficient computation of vietoris–rips persistence barcodes. *Journal of Applied and Computational Topology*, 5(3):391–423.

Bhat, P., MM, M. P., and Pai, R. M. (2023). Anomaly detection using federated learning: A performance based parameter aggregation approach. In *2023 3rd International Conference on Intelligent Technologies (CONIT)*, pages 1–6. IEEE.

Carrière, M., Chazal, F., Ike, Y., Lacombe, T., Royer, M., and Umeda, Y. (2020). Perslay: A neural network layer for persistence diagrams and new graph topological signatures. In *International Conference on Artificial Intelligence and Statistics*, pages 2786–2796. PMLR.

Che, M., Galaz-García, F., Guijarro, L., Membrillo Solis, I., and Valiunas, M. (2024). Basic metric geometry of the bottleneck distance. *the Proceedings of the American Mathematical Society*.

Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H. W., Sutton, C., Gehrmann, S., et al. (2023). Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

Fang, M., Cao, X., Jia, J., and Gong, N. (2020). Local model poisoning attacks to {Byzantine-Robust} federated learning. In *29th USENIX security symposium (USENIX Security 20)*, pages 1605–1622.

Gao, W., Guo, S., Zhang, T., Qiu, H., Wen, Y., and Liu, Y. (2021). Privacy-preserving collaborative learning with automatic transformation search. In *the Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 114–123.

Kairouz, P., McMahan, H. B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A. N., Bonawitz, K., Charles, Z., Cormode, G., Cummings, R., et al. (2021). Advances and open problems in federated learning. *Foundations and trends® in machine learning*, 14(1–2):1–210.

Ma, X., Zhou, Y., Wang, L., and Miao, M. (2022). Privacy-preserving byzantine-robust federated learning. *Computer Standards & Interfaces*, 80:103561.

Ma, Z. and Gao, T. (2024). Federated learning backdoor attack detection with persistence diagram. *Computers & Security*, 136:103557.

McMahan, B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. (2017). Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR.

Sun, T., Li, D., and Wang, B. (2022). Decentralized federated averaging. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(4):4289–4301.

Wang, H., Sreenivasan, K., Rajput, S., Vishwakarma, H., Agarwal, S., Sohn, J.-y., Lee, K., and Papailiopoulos, D. (2020). Attack of the tails: Yes, you really can backdoor federated learning. *Advances in Neural Information Processing Systems*, 33:16070–16084.

Xie, Y., Fang, M., and Gong, N. Z. (2024). Poisonedfl: Model poisoning attacks to federated learning via multi-round consistency. *arXiv preprint arXiv:2404.15611*.

Xiong, Z., Cai, Z., Takabi, D., and Li, W. (2021). Privacy threat and defense for federated learning with non-iid data in aiot. *IEEE Transactions on Industrial Informatics*, 18(2):1310–1321.

Xu, J., Jiang, Y., Fan, H., and Wang, Q. (2024). Svfldetector: a decentralized client detection method for byzantine problem in vertical federated learning. *Computing*, pages 1–21.