University of Zurich^UZH

MASTER THESIS – Communication Systems Group, Prof. Dr. Burkhard Stiller

Novel Topology Inference Attacks on DFL

Yuanzhe Gao
Zurich, Switzerland
Student ID: 20-752-200

Supervisor: Chao Feng, Dr. Alberto Huertas Celdran
Date of Submission: October 15, 2024

University of Zurich
Department of Informatics (IFI)
Binzmühlestrasse 14, CH-8050 Zürich, Switzerland

ifi

Master Thesis
Communication Systems Group (CSG)
Department of Informatics (IFI)
University of Zurich
Binzmühlestrasse 14, CH-8050 Zürich, Switzerland
URL: http://www.csg.uzh.ch/

Declaration of Independence

I hereby declare that I have composed this work independently and without the use of any aids other than those declared (including generative AI such as ChatGPT). I am aware that I take full responsibility for the scientific character of the submitted text myself, even if AI aids were used and declared (after written confirmation by the supervising professor). All passages taken verbatim or in sense from published or unpublished writings are identified as such. The work has not yet been submitted in the same or similar form or in excerpts as part of another examination.

Zürich,

Signature of student

Abstract

Decentralized Federated Learning (DFL), a new paradigm in the field of machine learning, reaches the goal of collaboratively training advanced models while preserving local data privacy by allowing multiple participants to forward model parameters to each other. Due to its decentralized nature, the network topology can be diverse. Different topologies have a significant impact on the performance and security of DFL. Therefore, this work proposes a novel topology inference attack specifically for DFL networks. This attack can be divided into two scenarios based on the level of information the attacker has: (1) In the first scenario, the attacker has all the node information and some edge information, and predicts the connection status of unknown edges by training a supervised machine learning model; (2) In the second scenario, the attacker only has all the node information and determines whether different nodes in the network are connected by using a clustering model from unsupervised learning framework. In addition, this work constructs four effective DFL network node metrics to facilitate these topology inference attacks.

The designed attack methods of this work have been evaluated in various DFL environments and achieves good performance in most of them, thus providing new insights for future securiy research in the field of DFL network.

Zusammenfassung

Dezentrales föderiertes Lernen (DFL), ein neues Paradigma im Bereich des maschinellen Lernens, erreicht das Ziel, fortgeschrittene Modelle gemeinsam zu trainieren und dabei den lokalen Datenschutz zu wahren, indem mehrere Teilnehmer Modellparameter aneinander weitergeben können. Aufgrund seines dezentralen Charakters kann die Netzwerktopologie vielfältig sein. Unterschiedliche Topologien haben einen erheblichen Einfluss auf die Leistung und Sicherheit von DFL. Daher wird in dieser Arbeit ein neuartiger Topologie-Inferenzangriff speziell für DFL-Netzwerke vorgeschlagen. Dieser Angriff kann je nach Informationsstand des Angreifers in zwei Szenarien unterteilt werden: (1) Im ersten Szenario verfügt der Angreifer über alle Knoteninformationen und einige Kanteninformationen und sagt den Verbindungsstatus unbekannter Kanten durch Training eines überwachten maschinellen Lernmodells voraus; (2) im zweiten Szenario verfügt der Angreifer nur über alle Knoteninformationen und bestimmt mithilfe eines Clustering-Modells aus einem Rahmen für unüberwachtes Lernen, ob verschiedene Knoten im Netzwerk verbunden sind. Darüber hinaus werden in dieser Arbeit vier effektive DFL-Netzwerkknotenmetriken erstellt, um diese Topologie-Inferenzangriffe zu erleichtern.

Die in dieser Arbeit entwickelten Angriffsmethoden wurden in verschiedenen DFL-Umgebungen evaluiert und erzielen in den meisten von ihnen eine gute Leistung, wodurch neue Erkenntnisse für die zukünftige Sicherheitsforschung im Bereich der DFL-Netzwerke gewonnen werden.

Acknowledgments

First of all, I would like to express my sincere gratitude to supervisor Chao Feng and Dr. Alberto Hueartas of my master thesis for their guidance. Especially Chao Feng, whose weekly meeting communication and his selfless help anytime and anywhere provided me with great motivation and help to complete this work. In addition, I am also very grateful to Prof. Dr. Stiller for allowing me to carry out this project at the Communication Systems Group. It was with this help that I was able to finish the work.

Contents

Chapter 1

Introduction

In recent years, machine learning (ML) has achieved significant progress in various fields. These successes are largely attributed to the availability of large-scale datasets and powerful computing resources. However, traditional machine learning models usually require centralized data collection, which means that individual devices or entities must share raw data with a central server for training purposes. This approach raises several issues, especially in terms of data privacy, security, and regulatory compliance, especially in industries such as healthcare and finance that involve sensitive information. These limitations have prompted the development of new paradigms designed to overcome these challenges.

Federated learning is seen as a potential solution to make up for the shortcomings of traditional machine learning. It allows multiple devices or nodes to cooperatively train a shared model without having to share local data, a feature that is achieved by retaining data on local devices, only model updates are sent to the central server for implementation, thereby enhancing privacy protection. Traditionally, this federated learning framework, known as centralized federated learning (CFL) , relies on a central server to orchestrate the training process, including collecting and distributing updated models to nodes. However, this centralized architecture has drawbacks such as single point failure and communication and computing bottlenecks. The proposed decentralized federated learning (DFL) avoids the dependence on the central server, which makes it possible for the nodes to communicate and cooperate in a completely decentralized manner. This model not only reduces the risk of centralization, but also improves the scalability and robustness of the training process.

1.1 Motivation

The rise of machine learning, particularly in privacy-sensitive domains such as healthcare, finance, and personal devices, has led to an increased focus on decentralized learning frameworks. Decentralized Federated Learning (DFL), a variant of FL, allows multiple participants to collaboratively train models while keeping their data localized. This decentralization not only enhances privacy by preventing data from being centralized but also improves the robustness of the system by distributing model updates across nodes.

However, DFL systems introduce new vulnerabilities, particularly related to network topology and communication patterns. While the primary focus of research in FL has been on protecting the data privacy of participants, less attention has been given to the privacy of the network structure itself. *Topology inference attacks*, where adversaries attempt to infer the underlying overlay graph of the DFL network, pose a serious risk. By mapping out the topology, adversaries can exploit the structure for further attacks, such as targeting specific nodes or manipulating model updates.

The motivation for this research arises from the need to explore and address this under-explored vulnerability. Understanding how topology inference attacks can be carried out in a DFL network is crucial for developing more secure and robust decentralized learning systems. This work aims to fill this gap by designing a novel topology inference attack targeting DFL network specifically and evaluates its performance across variaous domains.

1.2 Description of Work

This work designs a topology inference attack applicable to DFL networks, and evaluates the effectiveness of this attack in different DFL network environments. The reasons for the success of this attack are also analyzed, providing new insights into DFL network security.

The work begins by analyzing the different types of topology inference attacks from other network domains, and learn some successful experience from their topology inference attacks. Then, the topology inference attacks in DFL are divided into two different attack scenarios based on the level of knowledge of the attacker. In the first scenario, the attacker has all the node information and some of the edge information, while in the second scenario, the attacker is restricted to only having node information. This classification method is conducive to a specific analysis of the different types of attack methods.

Furthermore,This study also provides a detailed evaluation of the methods in different DFL network environments under different attack scenarios, and summarizes the lessons learned from successes and failures to facilitate the improvement of this attack strategy in the future.

1.3 Thesis Outline

The structure of this thesis is as follows:

- **Chapter 2: Background** – This chapter provides an overview of Federated Learning (FL) and Decentralized Federated Learning (DFL), focusing on the key differences between centralized and decentralized models. At the same time, the background on inference attacks is also introduced to clarify different inference attacks categories and the relevant objective and methods

- **Chapter 3: Related Work** – This chapter draws on the format by browsing network topology attacks that exist in other fields and attempts to apply them to the unique network environment of DFL. The focus is mainly on the node metrics and attack models used by other fields.
- **Chapter 4: Model Information Diffusion Process** – This chapter mathematically models the diffusion of model information from each node in the DFL network using matrix eigenvalue decomposition. This helps lay the foundation for the next attack setting.
- **Chapter 6: Design of Attack** – This chapter first classifies the designed topological inference attacks into two attack scenarios based on the different levels of knowledge possessed by the attacker. Second, it studies and proposes four node metrics that assist in this attack and analyzes their rationality. Finally, specific method strategies are proposed for different attack scenarios.
- **Chapter 6: Evaluation** – This chapter evaluates the proposed attack across different domains.It also provides specific analysis and insights into the performance of attacks in different environments.
- **Chapter 7: Summary and Conclusion** – The thesis concludes by summarizing the key findings, contributions, and limitations of the work. Recommendations for future research directions are also discussed.

Chapter 2

Background

2.1 Federated Learning

Federated Learning (FL) has emerged as an effective approach to collaborative machine learning, allowing multiple participants to jointly train models without sharing raw data. This technique addresses growing privacy concerns and regulatory constraints in centralized data collection by enabling local data processing. Over time, various architectures of FL have been developed, each offering different trade-offs in terms of scalability, privacy, and communication efficiency. The following section explores the primary FL architectures, including Centralized Federated Learning (CFL), Decentralized Federated Learning (DFL), and hybrid approaches such as semi-decentralized FL.

2.1.1 Different Architectures

Federated Learning (FL) can be implemented in a variety of architectural models, each designed to accommodate different privacy, scalability, and communication needs. The three primary architectures that have been widely studied are Centralized Federated Learning (CFL), Decentralized Federated Learning (DFL), and semi-decentralized models, which offer hybrid approaches. Each architecture has distinct advantages and challenges that influence its applicability across various use cases.

Centralized Federated Learning (CFL) The traditional *Centralized Federated Learning (CFL)* model involves a central server that coordinates the training process across multiple clients[1]. In this architecture, each client trains a local model on its own data and periodically sends model updates—typically in the form of gradients or parameters—to the central server. The server then aggregates these updates, usually using algorithms such as Federated Averaging (FedAvg), and sends the updated global model back to the clients for the next round of training.

CFL is widely adopted due to its relatively simple coordination mechanism, as the central server manages synchronization, model aggregation, and communication between nodes. However, this architecture has significant limitations. The central server represents a *single point of failure*, which can compromise the entire system if it becomes overloaded, compromised, or unavailable. Additionally, while CFL improves data privacy by keeping raw data on the clients, the server can still be vulnerable to inference attacks that attempt to reverse-engineer sensitive information from the shared model updates. Moreover, as the number of clients increases, communication bottlenecks and scalability issues can arise, further limiting the effectiveness of the CFL model in large-scale systems.

Decentralized Federated Learning (DFL) To overcome the inherent limitations of centralization, *Decentralized Federated Learning (DFL)* has been proposed as a more robust and scalable alternative. In DFL, there is no central server; instead, each client, or node, communicates directly with a subset of other nodes in the network[2]. These nodes exchange model updates locally, and the aggregation process occurs in a peer-to-peer manner, without the need for a global coordinator. The absence of a central server in DFL significantly enhances the system's *robustness*, eliminating the risks associated with a single point of failure. DFL also improves *scalability*, as nodes only need to communicate with a limited number of peers, reducing the overall communication overhead. However, DFL introduces new challenges. Without centralized coordination, the consistency of the global model becomes more difficult to maintain, as different nodes may receive updates asynchronously or at different rates, potentially affecting the convergence of the global model. Moreover, the decentralized communication patterns expose the system to new types of security threats, such as *topology inference attacks*, where adversaries attempt to infer the network structure by analyzing communication flows between nodes. This architecture is the primary focus of this research, as it offers both significant advantages and distinct vulnerabilities in the context of Federated Learning.

Semi-Decentralized Federated Learning A hybrid approach, known as *Semi-Decentralized Federated Learning*, seeks to combine the benefits of both centralized and decentralized models[1]. In this architecture, multiple local servers, or coordinators, manage clusters of clients, with each local server handling model aggregation for its respective cluster. These local servers may then communicate with a central server or directly with each other to perform a second layer of aggregation, depending on the specific design of the system.

Semi-decentralized FL addresses some of the *scalability and coordination issues* of fully decentralized models by reintroducing limited centralization at a local level. It can also reduce communication costs compared to CFL, as clients within a cluster can exchange updates locally, minimizing the load on the central server. However, this architecture still retains some of the centralization risks, as the local servers themselves can become bottlenecks or targets for attacks. The semi-decentralized model offers a trade-off between the simplicity of CFL and the flexibility of DFL, making it an attractive option for scenarios where full decentralization is not feasible, but improvements in robustness and scalability are still desired.

2.1.2 DFL Workflow

Decentralized Federated Learning (DFL) differs fundamentally from the centralized approach by eliminating the need for a central server to coordinate the training process. In DFL, each node (or client) independently trains a local model on its data and collaborates with other nodes in a peer-to-peer fashion to update the global model. This decentralized communication architecture brings unique challenges and opportunities. The DFL workflow can be broken down into several key phases, described as follows:

1. Initialization At the beginning of the training process, each node in the DFL network initializes a local model. The model architecture is typically the same across all nodes, but each node's model parameters are initialized independently. This ensures that each node starts from the same baseline model structure, although the parameter values may differ. The number of participating nodes and the communication topology (i.e., which nodes can exchange updates) are predetermined, and this topology may vary based on the design of the network (e.g., ring, mesh, or random graph).

2. Local Training Each node then trains its local model on its private dataset. During this phase, each node performs multiple iterations of gradient descent (or another optimization algorithm) on its data. Since the data is not shared with any other nodes, the local models at this point reflect only the knowledge from their respective datasets. The number of local training steps taken at each node can vary based on computational capacity or communication constraints, but typically all nodes perform training in parallel to avoid delays.

3. Peer-to-Peer Communication Once the local training phase is completed, the nodes engage in communication with their peers. In DFL, this communication happens directly between nodes rather than through a central server. Each node shares its updated model parameters (or gradients) with a subset of other nodes in the network, based on the predefined communication topology. For example, in a ring topology, each node may exchange updates only with its immediate neighbors, whereas in a fully connected topology, each node could communicate with all other nodes. The communication phase is critical in DFL because the local model updates are aggregated through these peer exchanges. Depending on the system's design, aggregation can happen at the node level or through a more complex, multi-round process where updates propagate gradually through the network.

4. Model Aggregation After exchanging model updates with their peers, each node aggregates the received updates with its own local model. This aggregation can take different forms, but the most common approach is to compute a weighted average of the models received from neighboring nodes. The specific weighting can be based on various factors, such as the number of training samples at each node or the trustworthiness of the peer. The aggregation process ensures that each node's model benefits from knowledge distributed across the network, even though no raw data is exchanged. The aggregation phase is decentralized, meaning there is no global view of the model at any point during training. Instead, each node maintains a partial, localized view of the global model, which gradually becomes more accurate as more communication rounds are completed.

5. Synchronization and Convergence DFL networks often face challenges related to synchronization, as different nodes may operate at varying speeds or may be temporarily unavailable. To address this, many DFL systems implement a loose synchronization mechanism where nodes wait for a predefined number of peers to complete their local updates before initiating the next round of communication. This ensures that stragglers do not unduly delay the training process. Over multiple communication and aggregation rounds, the models at each node gradually converge toward a globally optimal model. The number of rounds required for convergence depends on various factors, including the communication topology, the diversity of the data across nodes, and the aggregation mechanism. Unlike CFL, where convergence is typically centralized and deterministic, convergence in DFL is more dynamic and depends on the decentralized interactions between nodes.

2.1.3 Impact of Network Topology on DFL

In Decentralized Federated Learning (DFL), the underlying network topology plays a crucial role in determining the system's efficiency, scalability, and security. The peer-to-peer nature of DFL means that nodes exchange model updates directly with their neighbors, and the structure of these communication patterns is defined by the network's topology. Different topological structures, such as fully connected, ring, mesh, or random graphs, can have varied effects on the overall performance and robustness of the system.

1. Convergence Speed and Communication Efficiency The communication topology directly impacts how quickly updates propagate throughout the network, which in turn affects the convergence rate of the global model. In a fully connected topology, each node communicates with every other node, ensuring rapid dissemination of updates. However, this comes at a high communication cost, making it impractical for large-scale networks. In contrast, a ring topology, where each node communicates only with its immediate neighbors, reduces communication overhead but significantly slows down the convergence process due to the slower spread of model updates across the network. Mesh and random graph topologies offer a balance between these extremes. Mesh networks allow nodes to connect with multiple peers, increasing the speed of information flow without incurring the high costs of full connectivity. Random graph topologies introduce some level of unpredictability in communication but often perform efficiently in large networks by connecting nodes in a more distributed manner. The choice of topology must therefore consider the trade-off between communication efficiency and convergence speed, with denser topologies achieving faster convergence at the cost of higher communication complexity.

2. Scalability and Resource Distribution As DFL networks scale to include more nodes, the impact of the topology becomes even more pronounced. Sparse topologies, such as ring or grid structures, scale well in terms of communication load, as each node only needs to interact with a few peers. However, these structures may introduce heterogeneity in the performance of different nodes, as some nodes might receive updates more frequently than others. On the other hand, dense topologies, while ensuring more uniform model updates, can create communication bottlenecks and increase the computational load on individual nodes, reducing the system's overall scalability.

3. Security and Vulnerabilities The network topology also significantly affects the system's security, particularly regarding vulnerabilities such as topology inference attacks. In topologies with concentrated communication paths (e.g., star or hub-and-spoke structures), an adversary could target key nodes to disrupt or manipulate the training process. Additionally, the patterns of communication in sparse topologies may be easier to observe and exploit, enabling adversaries to infer the underlying network structure by monitoring the flow of information between nodes. DFL's decentralized nature, while enhancing privacy in terms of data locality, exposes the system to attacks that exploit the very communication patterns designed to ensure collaboration. Understanding the impact of topology on security is therefore critical in designing robust DFL systems that can withstand such attacks. In particular, the ability to infer the network topology from observable communication behaviors poses a significant threat, as it allows adversaries to map the system and selectively target or compromise key nodes.

2.2 Inference Attack

Inference attacks are adversarial techniques aimed at extracting sensitive information from machine learning models or systems without direct access to the underlying data. These attacks exploit the way models process and return predictions, leveraging observable behaviors, side-channel information, or model outputs to infer properties about the data, model parameters, or system structure. As Federated Learning (FL) and, specifically, Decentralized Federated Learning (DFL) systems are increasingly used for their privacy-preserving capabilities, they are also becoming vulnerable to various forms of inference attacks. This section examines several common types of inference attacks, their objectives, and the strategies adversaries use to carry them out.

2.2.1 Different Types of Inference Attacks

Inference attacks can be categorized into several types, each defined by the adversary's objective and the methods employed. Below are five common types of inference attacks, discussed in terms of their attack objectives and strategies:

Membership Inference Attack

- **Attack Objective**: A membership inference attack seeks to determine whether a specific data sample was part of the training dataset used to train the model[3]. This type of attack is particularly dangerous in scenarios where the inclusion of data in the training set reveals sensitive information, such as in healthcare or finance, where merely knowing a data point was used can be highly private.
- **Attack Strategy**: The adversary uses the model's outputs—typically the prediction confidence scores or probabilities—to infer whether a particular data point was in the training set[4]. By analyzing differences in the model's behavior on training data versus unseen data, the attacker can make informed guesses about whether a specific instance was included. Methods such as thresholding the prediction confidence or using shadow models trained to mimic the target model's behavior are commonly employed to enhance the attack's accuracy[5].

Model Inversion Attack

- **Attack Objective**: Model inversion attacks aim to reconstruct input data or infer certain properties of the input based on the model's outputs[6]. These attacks pose a significant privacy risk in applications involving sensitive data, such as facial recognition or medical data, as they can reveal personal information based on the model's learned knowledge.
- **Attack Strategy**: In this type of attack, the adversary leverages the model's output predictions and uses iterative optimization techniques to generate input data that could have led to the observed outputs. The process involves adjusting candidate inputs until the predicted outputs from the model match the attacker's target output[7]. Over several iterations, the attacker can approximate the original input used during training.

Property Inference Attack

- **Attack Objective**: The objective of a property inference attack is to infer general properties about the training data, rather than identifying specific data points. This attack targets aggregate characteristics of the dataset, such as demographic distributions or the presence of certain attributes in the data.
- **Attack Strategy**: Adversaries exploit model outputs or gradients to infer global properties of the dataset. Attackers often employ auxiliary models that are trained to detect specific properties by observing how the model behaves. In a DFL context, adversaries may monitor how model updates from different nodes react to certain inputs, allowing them to infer shared properties across nodes' local datasets, such as the prevalence of certain diseases in healthcare data.

Attribute Inference Attack

- **Attack Objective**: A attribute inference attack seeks to compromise the training process by inserting malicious data into the training set, with the aim of skewing the model's behavior or output[8]. Although this attack focuses on degrading the model's performance, it can also facilitate inference attacks by making models more vulnerable to data extraction.
- **Attack Strategy**: The adversary injects carefully designed malicious data points into the training process, which are intended to influence the learning process in specific ways. By modifying the training data, the attacker can induce the model to produce biased predictions or targeted errors. These manipulated models may inadvertently leak information about the training data when queried, providing an opening for further inference attacks.

Each type of inference attack targets different aspects of machine learning systems, from identifying individual data points (as in membership inference) to uncovering general dataset properties (as in property inference) or reconstructing the model itself (as in model extraction). Understanding the objectives and strategies behind these attacks is essential for analyzing vulnerabilities in DFL systems and developing effective defenses.

2.2.2 Different Knowledge Levels of Attackers

The effectiveness and strategy of an inference attack are significantly influenced by the level of knowledge the adversary possesses about the target model or system. In general, attackers can operate under varying levels of knowledge, ranging from minimal information to complete access. Understanding these different levels helps to classify attacks based on the assumptions made about the adversary's access to data, model parameters, and system configurations. Below are three common knowledge levels that impact the design and execution of inference attacks.

Black-box Knowledge

- **Overview**: In a black-box scenario, the attacker has minimal information about the target model or system. The adversary can interact with the model by submitting queries and observing the corresponding outputs (e.g., predictions or confidence scores), but they have no direct access to the model's parameters, architecture, or the underlying training data.
- **Attack Strategy**: Black-box attackers rely on the model's observable outputs to gather information about its behavior. By carefully crafting inputs and analyzing the outputs, they can infer sensitive information indirectly. For example, in membership inference attacks, black-box attackers might compare the model's confidence scores on different inputs to determine if a particular data point was part of the training

set. In property inference, black-box attacks typically involve using auxiliary models or shadow models to learn the relationships between outputs and hidden properties of the training data.

- **Limitations**: While black-box attacks are practical in many real-world scenarios where access to model internals is restricted, they are generally less precise and effective compared to attacks where the adversary has more information. Attackers in this category must compensate for the lack of direct access by leveraging statistical and heuristic techniques to derive meaningful inferences.

White-box Knowledge

- **Overview**: In white-box scenarios, the attacker has full access to the model's parameters, architecture, and possibly even the training data. This level of access allows the attacker to exploit the internal structure of the model to perform highly targeted attacks.
- **Attack Strategy**: White-box attackers have a significant advantage, as they can inspect model gradients, parameters, and other internal properties directly. This access enables more precise and efficient attacks, such as model inversion and model extraction. In model inversion attacks, for instance, the attacker can directly optimize the input space to reconstruct sensitive training data using the model's learned parameters. Similarly, in model extraction, a white-box attacker can replicate the target model's architecture and parameters with high fidelity by leveraging full access to the model's internals.
- **Advantages**: White-box attacks are often more successful than black-box attacks due to the wealth of information available to the adversary. This high level of access makes white-box attacks particularly concerning, especially in scenarios where the model is deployed in environments where users or external systems can gain direct access to model files, such as in edge devices or machine learning-as-a-service platforms.

Gray-box Knowledge

- **Overview**: A gray-box scenario represents an intermediate level of knowledge, where the attacker has partial access to the model or system. This could include knowledge of the model's architecture or training process but without access to the actual parameters or raw data used in training.
- **Attack Strategy**: Gray-box attackers often combine elements of both black-box and white-box strategies. For example, they might know the general architecture of the target model (e.g., the number of layers, types of activation functions) and use this information to train a surrogate model, which they then use to launch black-box-style attacks. Alternatively, they may have access to some model parameters but not the entire system, allowing them to refine their attack with partial knowledge of the model's inner workings.

- **Use Cases**: Gray-box attacks are common in scenarios where the attacker is an insider with some degree of access to system documentation or implementation details, or where the model's general structure is publicly known but its parameters remain private. These attacks are more targeted and effective than black-box attacks, though less precise than white-box approaches.

Each knowledge level—black-box, white-box, and gray-box—presents different challenges and opportunities for adversaries carrying out inference attacks. The more knowledge an attacker has about the system, the more powerful and effective their attack can be. Understanding the adversary's knowledge level is critical in designing defenses, as more sophisticated attacks often require more robust mitigation strategies.

2.2.3 Mechanisms of Attack Success

The success of inference attacks relies on specific mechanisms that exploit vulnerabilities within machine learning models and systems. These mechanisms stem from the inherent characteristics of how models are trained, the way data is processed, and the information exposed through model predictions or updates. Understanding the underlying mechanisms that enable successful inference attacks is critical for designing more robust defenses. Below are several key mechanisms that play a pivotal role in the success of these attacks:

1. Overfitting and Model Generalization Overfitting is one of the primary factors that contribute to the success of inference attacks[9]. When a model overfits to its training data, it learns not just the general patterns but also the specific details of the training examples, making it easier for adversaries to infer sensitive information. Overfitted models exhibit higher confidence when predicting on training data than on unseen data, a vulnerability that is frequently exploited in membership inference attacks.

In well-generalized models, the gap between the model's performance on training and unseen data is smaller, reducing the attack surface. However, in practice, models often retain subtle details of the training data, which attackers can leverage, particularly in scenarios where training data is imbalanced or highly representative of certain classes.

2. Model Confidence Scores and Probability Distributions Machine learning models often output probability distributions over predicted classes, providing detailed information about the model's confidence in its predictions. These confidence scores are a major point of vulnerability, as they expose nuanced differences between how the model responds to training data versus unseen data.

Adversaries in membership inference and model inversion attacks exploit these differences by analyzing the probability distributions of model outputs. Higher confidence scores on training data can signal the presence of that data in the training set, while patterns in these scores can be used to reconstruct input features, enabling model inversion. The granularity of the model's output plays a crucial role—more detailed probability distributions tend to increase the attack surface.

3. Model Architecture and Hyperparameters The architecture of a model, including the number of layers, types of activation functions, and hyperparameters, can inadvertently affect the model's vulnerability to inference attacks. Deep models with large capacity are more likely to memorize training data, increasing the risk of membership inference and model inversion attacks. Furthermore, hyperparameters such as batch size and learning rate can influence how much information is leaked through gradients or model outputs.

Attackers can also exploit knowledge of the model's architecture to conduct model extraction attacks. By knowing or guessing the architecture, adversaries can tailor their queries or reconstruction techniques to replicate the target model more accurately, enabling further inference attacks.

The success of inference attacks depends on exploiting the vulnerabilities in model training, data processing, and system design. Factors such as overfitting, confidence scores, gradients, model architecture, and communication patterns provide adversaries with the means to infer sensitive information from machine learning systems. Recognizing these mechanisms is essential for understanding how attacks succeed and for developing strategies to mitigate these vulnerabilities in decentralized learning environments.

Chapter 3

Related Work

This chapter provides an in-depth discussion of various research methods related to network topology inference. While this issue has received relatively little attention within DFL networks, extensive studies have been conducted in other network fields focusing on problems like link prediction and graph reconstruction. The chapter begins by summarizing approaches to topology inference in areas such as computer networks, social networks, and gene regulatory networks, with the aim of extracting insights applicable to the analysis of DFL networks. In addition, the characteristics of network data from different domains are also examined, highlighting both the similarities and differences compared to DFL network data.

3.1 Computer Network Domain

3.1.1 Network Characteristics

A computer network usually refers to a system in which multiple computing devices are connected to each other through links to transmit and share information[10]. The system consists of two basic modules: **nodes (network devices)** and **links**. Links connect multiple nodes together by defining how they carry information through communication protocols. Participants in a computer network can be broadly divided into two categories:

- **External nodes**: These nodes are located in the external area of the network and are terminal devices that users can directly control and access, such as personal computers and servers. External nodes are usually the source or destination of communication in the network[11].
- **Intermediate nodes**: These nodes sit between external nodes and are responsible for transmitting and relaying network components such as routers, switches, or network management devices. Intermediate nodes are usually invisible to the user and cannot be directly controlled or accessed[12].

Topology inference in computer networks generally refers to the inference of the link state and the number and structure of intermediate nodes under the premise of controlling part or all of the external nodes[13]. This kind of research is of great significance for improving the overall performance of the network and diagnosing abnormal nodes.

In addition, another important feature of computer networks is the ability to conduct directional communication between different external nodes. For example, an external device can send signals to reach another target external device and collect network performance measurement data through these signals (such as pinging a remote server through a personal computer). This directional communication feature is significantly different from many network types in other fields and has a profound impact on the topology inference methods unique to computer networks.

Therefore, in order to facilitate the analysis of topology inference problem, the most important characteristics of computer network data can be summarized as follows:

- **Hierarchical Nature of Node Roles:** The distinction between external nodes and intermediate nodes determines the scope and goal of topology inference, which usually relies on controllable information from external nodes to infer the overlay structure of the entire network;
- **Controllability of Directional Communication:** Directed communication between external nodes makes it possible to obtain network performance data between specified nodes through signal exchange. This feature provides an important basis for inferring network topology.

3.1.2 Inference Methods

Traceroute-Based Approach Traceroute is a widely used network diagnostic tool to detect the nodes on the route between a pair of source and destination nodes and measure the traffic delay from the source node to each intermediate relay node on the route[14]. It detects the network path from the source node to the destination node hop by hop by sending ICMP packets with increasing TTL (time to live), and obtains the IP address of the intermediate router. Each detection packet is sent by the source node, stops when it reaches the destination node, and relies on the intermediate nodes on the path for forwarding. Through this hop-by-hop detection mechanism, the internal path structure connecting these two external nodes can be estimated effectively.

In order to improve the performance of Traceroute in complex networks, researchers have proposed many improvement schemes, especially in the application of network topology inference. For example, the Rocketfuel method proposed by Spring et al.[15] greatly improves the accuracy and efficiency of inferring physical topology in Internet service provider (ISP) networks by optimizing Traceroute measurement paths and filtering unnecessary path information. In addition, Luckie[16] developed the Scamper tool, which can perform various types of active detection tasks in large-scale networks and expand the application of Traceroute in Internet topology measurement. These improvements not only enhance the applicability of Traceroute in large-scale complex networks, but also lay the foundation for more accurate network topology inference.

Compared to tomography-based approach, Traceroute's advantage are its simplicity and low-cost. However, its disadvantage is that when some routers do not respond to ICMP detection packets or when anonymous routers are included in the path, Traceroute's inference ability will be greatly limited[17]. In addition, it requires nodes to actively intervene in the communication within the network to obtain additional information, which is not feasible in many other types of networks, therefore it lacks portability in cross-domain applications.

Tomography-Based Approach The tomography-based method is a technology that relies on end-to-end measurement results to infer network topology[18]. It is often used in situations where it is impossible to directly obtain information about intermediate nodes, such as anonymous routers. This method collects link quality information (such as latency, packet loss rate, etc.) by sending probe packets, and infers the logical topology of the network based on statistical analysis. Unlike the Traceroute method, the Tomography method does not rely on the router's feedback on ICMP probe packets, but instead infers the relative distance between nodes and the routing tree structure by analyzing the communication measurement data between multiple terminal nodes.

In a typical tomography framework, the source node sends probe packets to multiple target nodes, which record the link quality (such as latency or packet loss rate) of the multi-hop path and return that information to the source node. Based on this information, the source node calculates the relative distances between each terminal node and uses these distances to build the logical tree structure of the network. The key challenge of the Tomography method is to choose the right distance metric to better reflect the relative distance between paths and meet the certain additivity requirements.

In order to improve the accuracy of tomography inference, researchers have proposed a variety of inference methods based on different distance metrics. The following are some of the more well-known methods:

1. **Sandwich Probe-based Metric:** This method estimates the length of the common segment on the path by sending three back-to-back packets, with a larger packet sandwiched between two smaller packets, using the extra delay difference caused by the larger packet. This technique is particularly suitable for measuring the longest common route (LCR) between two target nodes, thus helping to reconstruct the network topology. Delay differences between small packets reflect differences in path structure.

2. **Loss Rate-based Metric[19]:** The quality of the path is inferred by measuring the packet loss rate of the probe packet. The packet loss rate is estimated by sending small data packets and observing whether they are successfully received.The higher the packet loss rate, the longer or more congested the path. By comparing the packet loss rates between paths, it is possible to infer which paths share the same

segments. The mathematical relationship between packet loss rate and distance between nodes is as follows:

$$d(D_i, D_j) = \log\left(\frac{Pr(X_i = 1)}{Pr(X_j = 1)}\right) - \log\left(\frac{Pr(X_i = 1, X_j = 1)}{Pr(X_i = X_j = 1)}\right)$$

where $Pr(X_i = 1)$ is the probability that the packet is successfully received at node D_i and $Pr(X_i = X_j = 1)$ is the joint probability that both nodes D_i and D_j successfully receive the packet.

3. **Delay Cumulant-based Metric:** Delay cumulants are used to estimate the length of a path based on the high-order statistical properties of the delay. By sending packets with timestamps and recording the delay at each destination node, high-order cumulants of the delay distribution are calculated. These cumulants reflect the shape of the delay distribution and are applicable to networks with large delay variations. The r-order cumulant of the delay $K(X)$ is defined as:

$$K(X) = \frac{\partial}{\partial\theta}\log E(e^{\theta X})\bigg|_{\theta=0}$$

where:

- $K(X)$: The r-order cumulant of the measured delay.
- X: The measured delay.
- $E(e^{\theta X})$: The expected value of the exponent of delay.

Compared with the traceroute method, the advantage of the tomography-based approach is that it does not rely on the cooperation of intermediate nodes to forward detection packets[20]. Its end-to-end measurement data generally conforms to the normal behavior of network communication protocols and can therefore be used in a wider range of computer networks. However, the Tomography method also has two significant drawbacks. First, it can only infer the logical topology, not physical ones, which may lead to information loss, such as failure to recognize multiple nodes located on the same branch. Secondly, the method assumes that the whole network has a tree-like structure. If there exists loops in the network, the connections between nodes will be more diverse, and the analysis cannot be carried out simply by the longest common branch (LCR) method, and the complexity of topology inference will increase significantly. Although the assumption of a tree structure is reasonable in many modern computer networks, the applicability of this approach may be limited greatly when generalized to other domains.

Chapter 4

Node Information Diffusion on DFL Network

1. Problem Context

Decentralized Federated Learning (DFL) involves multiple nodes (clients) collaboratively training models by performing local updates and sharing these updates with their neighbors. Unlike traditional federated learning, there is no central server to aggregate the model parameters. Each node communicates directly with its neighbors and aggregates their updates in a decentralized manner.

The goal of this mathematical modeling is to analyze the convergence properties of this DFL process under a synchronized setup, where all nodes perform aggregation and local updates simultaneously in discrete rounds.

2. Network Representation

The network is represented as a graph $G = (V, E)$, where:

- V is the set of nodes (clients).
- E is the set of edges representing direct communication links between nodes.
- The adjacency matrix A encodes the network structure, with $A_{ij} = 1$ if nodes i and j are connected, and $A_{ij} = 0$ otherwise.
- The degree matrix D is a diagonal matrix where $D_{ii} = \text{degree}(i) + 1$ (including self-loops).

3. Aggregation Matrix P

The aggregation matrix P is defined as:

$$P = D^{-1}(A + I)$$

where:

- $A + I$ includes self-loops, allowing nodes to incorporate their own model parameters during aggregation.
- D^{-1} normalizes by the degree, ensuring that the sum of each row in P is 1, making it a stochastic matrix.

4. Model Parameter Representation

The model parameters at each node are represented as a matrix M_t at round t, where each row $\theta_i^{(t)}$ represents the model parameters of node i at round t:

$$M_t = \begin{bmatrix} \theta_1^{(t)} \\ \theta_2^{(t)} \\ \vdots \\ \theta_N^{(t)} \end{bmatrix}$$

5. DFL Process Over Multiple Rounds

The DFL process over T rounds is described by the following update equations:

1. **Initial State:** Each node starts with its own initial model parameters:

$$M_0 = \begin{bmatrix} \theta_1^{(0)} \\ \theta_2^{(0)} \\ \vdots \\ \theta_N^{(0)} \end{bmatrix}$$

2. **Aggregation Step:** At each round t, each node aggregates model parameters from its neighbors:

$$\tilde{M}_t = PM_{t-1}$$

where $\tilde{M}_t$ represents the aggregated model parameters before local updates.

3. **Local Update Step:** Each node then updates its aggregated model parameters using its local dataset:
$$M_t = \tilde{M}_t + \delta_t = PM_{t-1} + \delta_t$$
where δ_t represents the vector of local updates for each node in round t.

4. **General Update Equation:** Recursively, the model parameters at each round can be written as:
$$M_T = P^T M_0 + P^T \delta_1 + P^{T-1}\delta_2 + \cdots + P\delta_T$$
This equation captures the cumulative effect of the initial model parameters M_0 and the local updates δ_t over T rounds of aggregation and local training.

4.1 Key Assumptions

1. **Network Connectivity:** The graph G is connected, ensuring that information can propagate across the entire network.

2. **Stochastic Matrix:** The matrix P is row-stochastic, meaning that each row sums to 1. This ensures that the influence of neighboring nodes is appropriately normalized.

3. **Spectral Radius:** The spectral radius $\rho(P) < 1$. This ensures that P^T decays as T increases, indicating diminishing influence of initial model parameters over time.

4. **Bounded Local Updates:** The local updates $\delta_i^{(t)}$ are bounded, i.e., there exists a constant $C > 0$ such that for all t and nodes i:
$$\|\delta_i^{(t)}\| \leq C.$$
This condition ensures that local updates do not dominate the aggregation term in the long run.

4.2 Dominance of the Aggregation Matrix

This section aims to show that the terms involving P in the DFL update equation play a dominant role in determining M_T.

4.2.1 Initial Model Influence

The term $P^T M_0$ represents the influence of the initial model parameters after T rounds of aggregation. Given the spectral radius $\rho(P) < 1$, the norm $\|P^T\|$ decays exponentially:
$$\|P^T M_0\| \leq \|P^T\|\|M_0\| \to 0 \quad \text{as} \quad T \to \infty.$$
This implies that the effect of the initial model parameters diminishes over time. However, in the initial rounds, $P^T M_0$ can still have a significant influence due to the dominant eigenvalue of P.

4.2.2 Local Update Influence

The cumulative effect of local updates is given by the series:

$$\sum_{t=1}^{T} P^{T-t+1}\delta_t.$$

Each term $\|P^{T-t+1}\delta_t\|$ is bounded by:

$$\|P^{T-t+1}\delta_t\| \leq \|P^{T-t+1}\|\|\delta_t\|.$$

Since $\|\delta_t\| \leq C$ and $\|P^{T-t+1}\| \to 0$ as $T \to \infty$, the influence of each individual local update diminishes over time.

4.2.3 Comparative Dominance

In the initial rounds, the aggregation term $P^T M_0$ dominates because the local updates have not accumulated significantly. As T increases, the terms $\|P^{T-t+1}\|$ decay, causing the local updates to have a limited impact. The series converges if:

$$\sum_{t=1}^{T} \|P^{T-t+1}\| < \infty.$$

This convergence condition is met due to the spectral radius condition $\rho(P) < 1$.

4.3 Mathematical Conclusion

The aggregation matrix P and its powers P^t dictate the model evolution, especially in the initial rounds. The decaying influence of P^t over time ensures that the model parameters converge to a consensus value. This is largely influenced by the network topology.

As the local updates are bounded and diminish in influence due to the decaying powers of P, the final model parameters at all nodes will be determined by the weighted average of initial parameters, expressed by the network topology. Thus, the network structure plays a dominant role in the convergence behavior of the DFL process.

Chapter 5

Attack Design

Building on the understanding of topology inference methods from other network domains and the information diffusion process in DFL networks, this chapter proposes attack methods specifically targeting the topology of DFL networks. First, different attack scenarios are defined based on the level of information available to the attacker. Next, a series of node metrics are selected as the foundation for the attack, considering the characteristics of DFL networks, and their effectiveness is preliminarily assessed. Finally, detailed strategies are developed for each attack scenario, along with the necessary models and algorithms to facilitate implementation and evaluation.

5.1 Attack Specification

Although the specific operation of DFL is thoroughly explained in Chapter 2, a graph model is still adopted to provide a concise representation of the DFL network, aiming to clearly illustrate the various elements of the attack. Assume that the network is represented as $G = (V, E)$, where Y denotes the adjacency matrix of size $|V| \times |V|$. Based on the characteristics of the DFL network, the following assumptions are made for G:

1. **Undirected:** Since data transmission between nodes is bidirectional during the model aggregation process, G is treated as an undirected graph.
2. **Unweighted:** Given that the data distribution between nodes is independent and identically distributed (IID), G is considered an unweighted graph. Consequently, Y is a binary matrix.
3. **Static:** Before all rounds are completed, the connections between nodes remain unchanged, so G is regarded as a static graph.

For the vertex information in G, there exists a set of node metrics related to the network system, which is denoted as $x = (x_1, \ldots, x_{|V|})^T$. As for the edge information, binary indicators $y = [y_{ij}]_{1 \le i < j \le |V|}$ represent entries in the adjacency matrix Y. With this setting, the attacker's goal in a topological inference attack can be defined as follows: Given x and y, find an inferred adjacency matrix Y' as close as to the Y.

Attacker Capability: The attacker is assumed not to actively interfere with the default communication protocols and methods of DFL network G. In other words, the attacker is prohibited from sending non-compliant signals such as probe packets from controlled nodes to probe the location of other nodes.

Attack Scenarios: Depending on the amount of information the attacker has about the nodes and edges, the topological inference attacks in DFL networks can be divided into the following three scenarios. In these three scenarios, the a priori knowledge the attacker has is gradually reduced, making the attack more difficult and more realistic. Considering that for node metrics x and edge information y, it is often not easy for an attacker to obtain all the information, the two are split into $x = \left[x^{\text{known}}, x^{\text{missing}}\right], \quad y = \left[y^{\text{known}}, y^{\text{missing}}\right].$

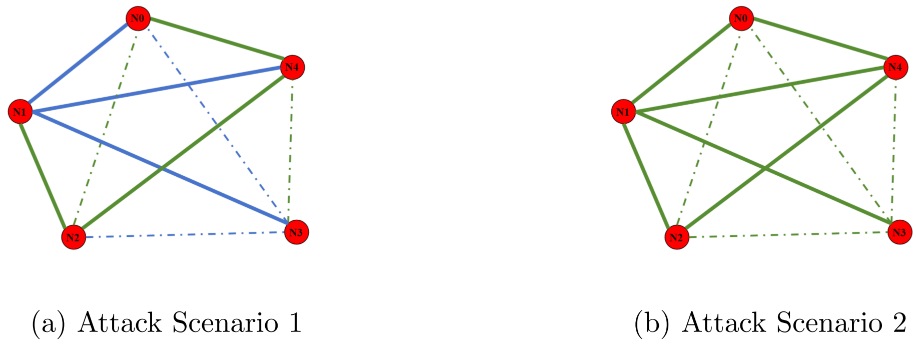
1. **Full Node Knowledge with Partial Edge Information** In this case, the attacker by default has all the relevant information about the nodes in the DFL network, such as model parameters, training datasets, etc., and thus can also get the node metrics of each node. In addition to this, the attacker also knows information about some of the edges in the network, i.e., the connectivity of some of the nodes. Therefore, the attacker's attack in this scenario can be understood as reasoning through all the node metrics and partial edge information to derive the missing links.

   This attack scenario is a more white box situation because the level of knowledge possessed by the attacker is quite broad and the difficulty of inference is not very high - only a portion of edge information.

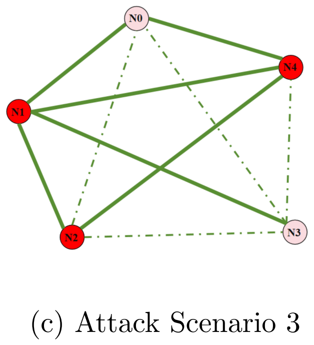
2. **Partial Node Knowledge with No Edge Information** The attacker in this case is considered to have only information about all the nodes and no longer has any knowledge about the edges in the network. In this case, the attacker's knowledge is further limited, approaching the gray-box state described in nearly half of the inference attacks.

3. **Partial Node Knowledge with Partial Edge Information** This is the most stringent attack scenario under our definition. Although the attacker has partial information about the edges, he cannot construct the known node-edge correspondence due to the lack of information about some nodes. This is very different from the attack method in the first scenario. Therefore, it is considered to be the most recent attack scenario for black boxes.

In this work, the main considerations are the first two scenarios, and some methodological assumptions are made for the third scenario as appropriate. Figure 5.2 provides a vivid visualization of the different knowledge information possessed by attackers under different attack scenarios. Among them, the edges with blue edges represent the edge set in the real network topology, while the green ones represent the non-edge set; the solid edges represent the information about the known edges, while the dashed edges represent the information about the unknown edges; the red nodes represent the nodes that the attacker knows, while the pink ones represent the nodes that cannot be controlled.



(a) Attack Scenario 1 (b) Attack Scenario 2



(c) Attack Scenario 3

Figure 5.1: Visualization of the Known and Unknown Information of Different Attack Scenarios

5.2 Node Metrics of DFL Network

Since the level of knowledge possessed by the attacker varies in different scenarios of DFL topology inference attacks, it is a reasonable choice to design different attack strategies fro each scenario. However, in either scenario, the attacker's control over the node metric information is indispensable. Therefore, **the first step in designing a topology inference attack should be to find appropriate node metrics that matches the characteristics of the DFL network**.

5.2.1 Potential Node Metrics

The quality of node metrics plays a vital role in the success of topology inference attacks in networks. They not only reflect the attributes and roles of each node in the network, providing essential data for developing attack strategies but also capture the relationships between the relative positions of nodes, serving as a bridge between the known node information and the inference of the unknown edges that the attacker seeks to uncover. A high-quality node metric should satisfy the following two conditions:

1. First, the feature data it represents should be easy to obtain and compute so as to avoid adding extra complexity to the attack method.

2. Second, the node metric should be able to effectively reflect the structural characteristics of the network, making it easy to infer the connectivity between nodes.

For example, in social networks, commonly used node metrics include the number of mutual friends and the frequency of interactions, which can help reveal underlying connectivity relationships, while in computer networks, communication packet loss and latency are classic topology inference metrics. In DFL network domain, since it is essentially a process in which multiple participants jointly train machine learning models under a decentralized structure, the choice of node metrics should focus on the relevant information about the models trained by each node.

Based on this logic, this work proposes four unique node metrics applicable to DFL networks and defines their specific calculation methods:

- **Relative Loss.** This metric measures the difference in loss when a locally trained model on one node is tested on the dataset of another node. It aims to reflect the generalizability of models trained on different nodes to other nodes. In general, the closer the relative position of two nodes in the DFL network, the higher the aggregation frequency between them, which makes the generizability of these two models better. A high relative loss indicates poor generalization and high overfitting level. It is computed by the following expression:

$$\text{Relative Loss}_{i,j} = \mathcal{L}(f_i, D_j) \tag{5.1}$$

  where D_i represents the training dataset of node i and f_i represents local model trained on node i. $\mathcal{L}(f, D)$ is the loss function used by trained model.

- **Relative Entropy.** Like relative loss, it is defined as the entropy from one node's local model to another node's training set. It measures the uncertainty in the model's predictions, providing insight into how confident the model is when predicting across different nodes' datasets. This node metric also tries to reflect the relative position relationship between nodes in the whole network from the perspective of their model generizability. However, unlike loss, entropy captures the confidence of the model but does not necessarily reflect accuracy. A model can be highly confident (low entropy) of its predictions but still perform poorly on other node's dataset. Therefore, it mainly plays a supplementary role to relative loss metric. The following is how it's calculated:

$$\text{Relative Entropy}_{i,j} = -\frac{1}{|D_j|} \sum_{x \in D_j} \sum_{k} y_k(x) \log(f_{i,k}(x)) \tag{5.2}$$

  where D_i represents the training dataset of node i and f_i represents local model trained on node i. $\mathcal{L}(f, D)$ is the loss function used by trained model.

Both of these node metrics aim to explore the connectivity between different pairs of nodes by analyzing the difference in performance of a node's model on its own training set compared to other nodes' datasets. As shown in Equation (5.1) and (5.2), whether for Relative Loss or Entropy, both require knowledge of the local models and training datasets of different nodes for calculation. In addition, due to the inherent randomness of machine learning model training, the values of these two metrics calculated from node i to node j are not equal to those calculated from node j to node i. In other words, these two pairwise node metrics are asymmetric. This asymmetry is particularly important when building attack strategies based on these metrics next.

The remaining two proposed node metrics focus on building connections based on the similarity relation of different node models. The relationship is defined from different vector space perspectives:

- **Cosine Similarity.** Cosine similarity measures the angular distance between the parameter vectors (weights) of two models[21]. In model comparison, it is mainly used to assess whether the parameter vectors are pointing in the same direction, regardless of their actual sizes. In the DFL network, when the models of two nodes have a large cosine similarity, it means that the two nodes aggregate with each other's model more frequently, thus showing a relatively close relationship between the two in terms of position. The formula for calculating this metric is as follows:

$$\text{Cosine Similarity}(\mathbf{a}, \mathbf{b}) = \frac{\mathbf{a} \cdot \mathbf{b}}{||\mathbf{a}|| \, ||\mathbf{b}||} \tag{5.3}$$

  where $\mathbf{a}$ and $\mathbf{b}$ represent two models from different nodes

- **Euclidean Distance.** Unlike Cosine Similarity, Euclidean Distance measures the straight-line distance between two vectors in multi-dimensional space, which reflects the numerical difference between the parameter vectors of the two models[22]. It is defined as follows:

$$\text{Euclidean Distance}(\mathbf{a}, \mathbf{b}) = ||\mathbf{a} - \mathbf{b}|| = \sqrt{\sum_{i=1}^{n} (a_i - b_i)^2} \tag{5.4}$$

  where $\mathbf{a}$ and $\mathbf{b}$ represent two models from different nodes

Unlike relative loss and entropy, these two model similarity metrics—cosine similarity and Euclidean distance—only require knowledge of the model parameters between nodes for calculation. In comparison, they do not depend on the training datasets of individual nodes, making them easier to obtain and use for constructing more potent topology inference attacks. Furthermore, these pairwise metrics are symmetric, as the model parameter vectors remain fixed during calculation. This symmetry simplifies the development of corresponding attack strategies.

5.2.2 From Node to Edge Information

Having identified a range of alternative node metrics that can be used in topology inference attacks on DFL networks, the next critical step is to establish how these node metrics relate to the topology, i.e., the information about the edges. There are many kinds of considerations for information about edges in a network, such as the distance, the number of paths between two nodes, or more intuitively whether two nodes are directly connected.

Distance is an inference objective that is often used in other network topology domains. It represents the number of edges in a shortest path connecting them. In general, when the distance between two nodes is smaller, these two nodes will be closer in the network. Using the distance between two nodes as the object that the node metrics speculate about seems to be a reasonable choice. For example, when the Relative Loss between node i and node j is very low, the distance between these two nodes is considered to be very short. But this correspondence between metrics faces two serious flaws:

1. **Distance metric is a continuous positive integer variable.** The distance value between two nodes can range from 0 to the diameter of this network. This provides a great challenge for inferring the number of distances between nodes undoubtedly, especially in the second attack scenario where only node information is available. Even if an attacker finds that some of node metrics exhibit similar characteristics, they still cannot be effectively labeled because of the large number potential distance results.

2. **Distance may not be accurately determined in networks with loops.** In a network like DFL with potential loops, the distance information between nodes may not be accurately reflected only by the partial edge information. In this case, the attacker's basis for issuing topology inference often lies in learning the relationship between node metrics and known edge information. However, having partial connectivity information does not always provide an accurate representation of the distance between nodes. For instance, nodes in a long chain may have shortcuts that significantly reduce the distance between them. Consequently, the adversary may infer incorrect patterns, which may deviate from the expected results of missing links.

Due to the influence of the two factors mentioned above, the distance between nodes is not suitable for establishing node metrics and topology (edge) information in DFL networks. Similarly, other commonly used metrics from different fields, such as centrality and the number of common neighbors, are also excluded. An ideal metric for representing edge information should have as few distinct values as possible while still being accurately calculable in various scenarios. Therefore, based on this requirement whether nodes are directly connected is proposed as an indicator variable, $\mathbb{I}_{\{i,j\}}$, as shown below to fulfill this requirement.

$$\mathbb{I}_{\{i,j\}} = \begin{cases} 1, & \text{if node } i \text{ and node } j \text{ are directly connected} \\ 0, & \text{otherwise} \end{cases}$$

Compared to other metrics, $\mathbb{I}_{\{i,j\}}$ has the following advantages: First, it is a binary identifier with only two elements in its value range, 0 and 1, which divides all nodes into two categories: direct link group and non direct link group. This greatly reduces the difficulty of the topology inference attack, transforming the inference task into a binary classification task. Second, This variable can also be obtained correctly with only partial edge information, and is not potentially misleading as distance can be. Finally, after knowing the value of the variable, it is easy to get the information about the edge that needs to be inferred, without having to perform a heuristic inference process like the distance does. This greatly facilitates the implementation of the attack.

5.2.3 Metrics Validation

Although the process of determining whether the two nodes are directly connected through alternative node metrics seems to be convincing and convenient, it still requires preliminary validation to assess its rationality and determine whether it can be used as the main basis for DFL network topology inference attack. An intuitive way to verify this process is to show whether there is a difference in the node metrics of directly connected node pairs and indirectly connected node pairs in the DFL network. If there is a significant difference in the node metrics between these two groups, it means that this classification method can be used to study and analyze the relationship between node metrics and network topology. On the contrary, if the node metrics of the two groups are very similar, it means that in the DFL network, there is no significant difference between directly connected node pairs and non-directly connected node pairs, which further indicates that it is impossible or more difficult to use this binary classification method to determine the edge relationship between nodes.

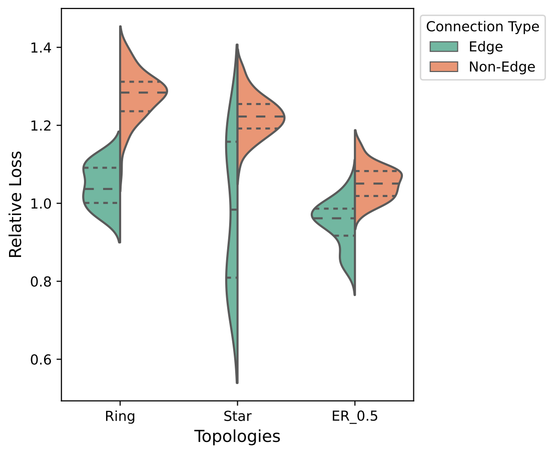
For this reason, the node metric distribution plot for these distinct two groups is selected as an effective visualization method to present this preliminary validation results. Based on Figure5.2 and Figure5.3, the following preliminary conclusions can be drawn: (1) For the two metircs related to model generizability, the difference between the two different groups is greater than the similarity metrics; (2) The cosine similarity values of the nodes are the closest, which indicates that the similarity between the models of the nodes is very high; (3) According to this preliminary distribution verification plot, the node metric - Relative Loss is expected to be an indicator of the best attack results.

5.3 Attack Strategies

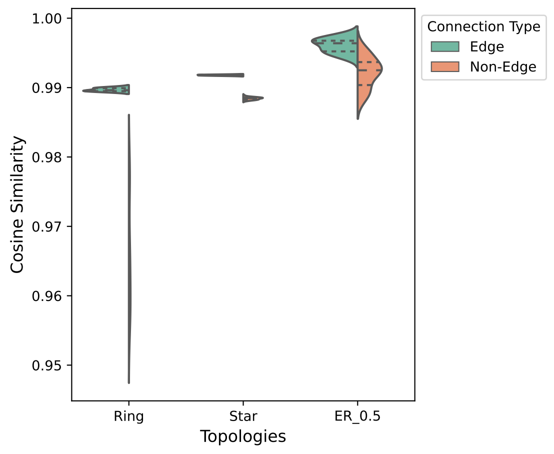
After verifying the effectiveness of the proposed node metrics for distinguishing whether there is a direct connection between pairs of nodes, specific attack strategies for topology inference attacks in DFL networks can be formulated based on this logic. A general framework for different attack scenarios is proposed and then based on this framework, some derivations of algorithms are also listed aiming at obtaining a comprehensive assessment of the effectiveness of the attack strategy.

5.3.1 General Process for Different Attack Scenario

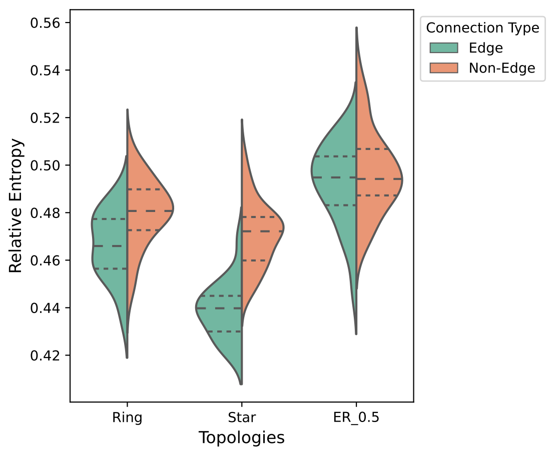
Considering that topology inference attacks in DFL are categorized into two main attack scenarios, it is a reasonable move to set up a specific attack strategy for each scenario.



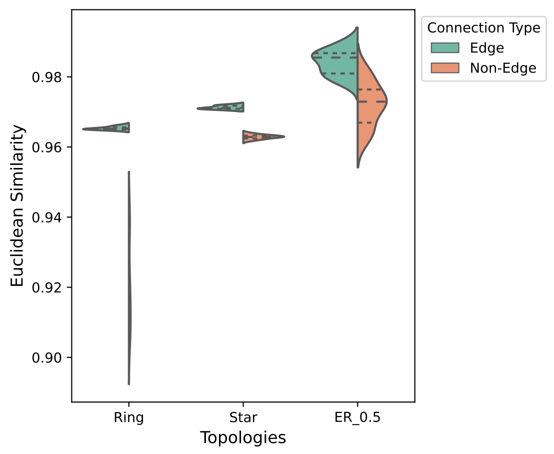
(a) Relative Loss



(b) Cosine Similarity

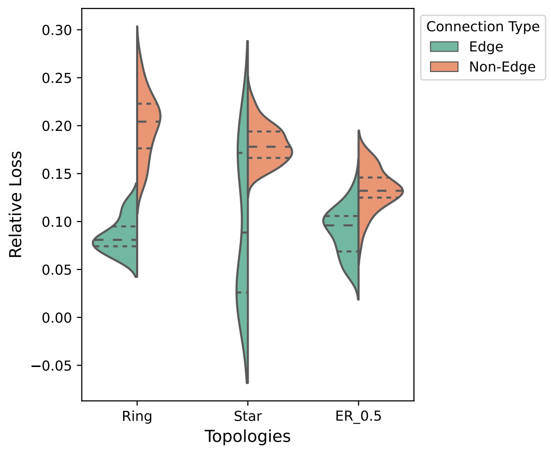


(c) Relative Entropy

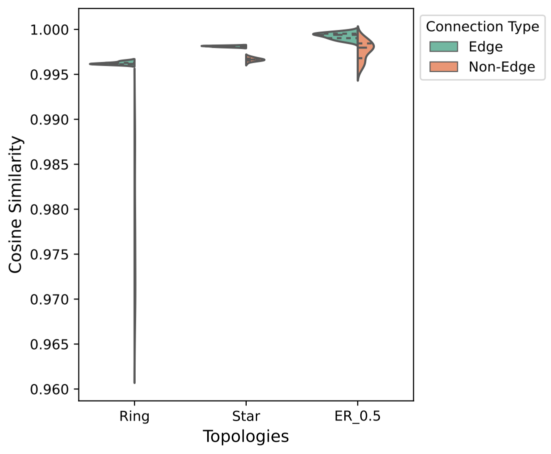


(d) Euclidean Distance

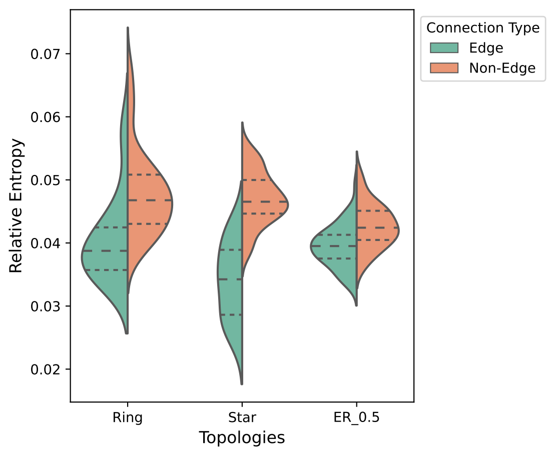
Figure 5.2: Different Node Metrics Distribution Comparison under CIFAR-10 Dataset



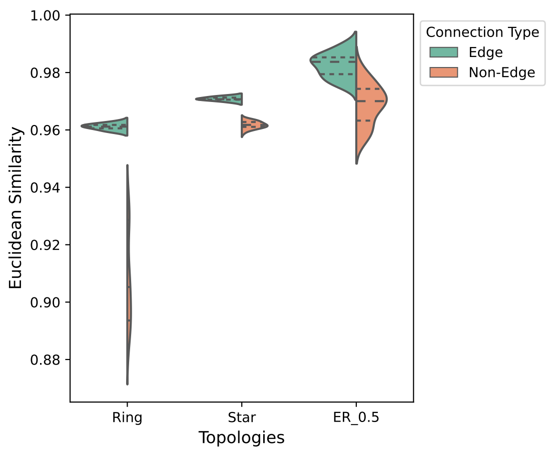
(a) Relative Loss



(b) Cosine Similarity



(c) Relative Entropy



(d) Euclidean Distance

Figure 5.3: Different Node Metrics Distribution Comparison under Mnist Dataset

Attack Scenario 1: Full Node Information with Partial Edge Information In this case, the attacker is considered to be in control of all the nodes' node metrics and a portion of the edges' information. The goal of the attacker is to speculate the information of the unknown part of the edges. Considering that both kinds of information are acquired, one reasonable attack strategy is to utilize **training a supervised machine learning model based on the known node information and edge information, and then use this trained model to predict the information of the unknown edges**. For the model to be trained, X in its input is the node metrics, while Y is the known information about the edges, labelled as 0 and 1. The process of model training can be thought of as learning the corresponding features of the known node metrics and edge information, which can be further applied to predict the connectivity between unknown pairs of nodes.

The general process of this attack strategy consists of three main phases: (1) Prepare training and test datasets, (2) Initialize Machine Learning Model, and (3) Predict missing results. The specific process is shown in the following Algo26.

Attack Scenario 2: Full Node Information with No Edge Information In this case, the attacker is assumed to have all the information about the nodes and none about the edges. Unlike the first attack scenario, because of the lack of edge information to be used as Y-values, it is not possible to construct a supervised model in order to learn features with known information. Therefore, **the attack strategy for this scenario is to use the clustering model in unsupervised learning to take the node metric of all nodes as input and divide them into two disjoint groups that are directly connected or not**.The advantage of this is that there are only two possible connectivity relationships between the nodes, so the attacker can explicitly go ahead and build two clusters to house different pairs of nodes respectively.

The general process of this attack strategy consists of three main phases: (1) Initialize unsupervised clustering model, (2)Fit the model to node metrics and get clusters, and (3) Mark each cluster with directly connected or not label based on the specific node metrics. The specific process is shown in the following Algo38.

Algorithm 1 Topology Inference Attack using Supervised Learning

```
Require: $X$: Node metrics, $Y^{known}$: Known edges, $Y^{missing}$: Missing edges
Ensure: $Y'$: Predicted adjacency matrix
1: function TopologyInference($X, Y^{known}, Y^{missing}$)
2:     $D_{train} \leftarrow$ ExtractNodePairsAndLabels($X, Y^{known}$)        ▷ Training data
3:     $D_{test} \leftarrow$ ExtractNodePairs($X, Y^{missing}$)        ▷ Test data (missing edges)
4:     $model \leftarrow$ InitializeSupervisedModel()        ▷ Initialize the ML model
5:     $model$.train($D_{train}$)        ▷ Train model on known edges
6:     $Y'^{missing} \leftarrow model$.predict($D_{test}$)        ▷ Predict missing edges
7:     $Y' \leftarrow$ Combine($Y^{known}, Y'^{missing}$)        ▷ Form the complete adjacency matrix
8:     return $Y'$
9: end function
10: function ExtractNodePairsAndLabels($X, Y^{known}$)
11:     $D_{train} \leftarrow []$
12:     for each edge $(i, j)$ in $Y^{known}$ do
13:         $node\_metrics \leftarrow GetNodeMetrics(X[i], X[j])$        ▷ Retrieve node metrics for pair $(i, j)$
14:         $label \leftarrow Y^{known}[i, j]$        ▷ Use known edge information as label
15:         $D_{train}.append((node\_metrics, label))$
16:     end for
17:     return $D_{train}$
18: end function
19: function ExtractNodePairs($X, Y^{missing}$)
20:     $D_{test} \leftarrow []$
21:     for each node pair $(i, j)$ in $Y^{missing}$ do
22:         $node\_metrics \leftarrow GetNodeMetrics(X[i], X[j])$        ▷ Retrieve node metrics for pair $(i, j)$
23:         $D_{test}.append(node\_metrics)$
24:     end for
25:     return $D_{test}$
26: end function
```

Algorithm 2 Topology Inference Attack using Unsupervised Clustering

```
Require: $X$: Node metrics, number of clusters $K = 2$ (Edge group and Non-edge group)
Ensure: $Y'$: Predicted adjacency matrix
 1: function TopologyInference($X$)
 2:     $clusters \leftarrow$ ClusterNodeMetrics($X, K = 2$)  ▷ Cluster node metrics into two groups (edge, non-edge)
 3:     $group_1, group_2 \leftarrow$ SeparateClusters($clusters$)  ▷ Separate node pairs into two clusters
 4:     $avg_1 \leftarrow$ CalculateMean($group_1$)  ▷ Calculate mean value of node metrics in group 1
 5:     $avg_2 \leftarrow$ CalculateMean($group_2$)  ▷ Calculate mean value of node metrics in group 2
 6:     if $avg_1 < avg_2$ then
 7:         $EdgeGroup \leftarrow group_1$
 8:         $NonEdgeGroup \leftarrow group_2$
 9:     else
10:         $EdgeGroup \leftarrow group_2$
11:         $NonEdgeGroup \leftarrow group_1$
12:     end if
13:     $Y' \leftarrow$ ConstructAdjacencyMatrix($EdgeGroup, NonEdgeGroup$)  ▷ Build adjacency matrix based on the groups
14:     return $Y'$
15: end function

16: function ClusterNodeMetrics($X, K$)
17:     $model \leftarrow$ InitializeClusteringModel($K$)  ▷ Initialize unsupervised clustering model (e.g., K-means)
18:     $clusters \leftarrow model.fit$(X)  ▷ Fit the model to node metrics and get clusters
19:     return $clusters$
20: end function

21: function SeparateClusters($clusters$)
22:     $group_1 \leftarrow$ [], $group_2 \leftarrow$ []
23:     for each $(i, j)$ in node pairs do
24:         if $clusters[i, j] = 1$ then
25:             $group_1.append(node\_metrics(i, j))$
26:         else
27:             $group_2.append(node\_metrics(i, j))$
28:         end if
29:     end for
30:     return $group_1$, $group_2$
31: end function

32: function ConstructAdjacencyMatrix($EdgeGroup, NonEdgeGroup$)
33:     Initialize adjacency matrix $Y'$ with all entries as 0
34:     for each $(i, j)$ in $EdgeGroup$ do
35:         $Y'[i, j] \leftarrow 1$  ▷ Mark edge group pairs as connected
36:     end for
37:     return $Y'$
38: end function
```

Chapter 6

Evaluation

6.1 Experiment Setup

6.1.1 Dataset and Models

All experiments are conducted on the following three datasets: MNIST, Fashion-MNIST, and CIFAR-10. These datasets are not only used as benchmarks in the field of model training but are also widely referenced in Inference Attack-related literature. Figure shows the grayscale and RGB visualizations of these datasets.

- **MNIST[23]** is the standard dataset for handwritten digit classification and contains 60,000 training set images and 10,000 test set images. Each image has a resolution of 28x28 pixels and is in grey scale, with ten classes of digits ranging from 0 to 9. A Multilayer Perceptron (MLP) is employed for this dataset's training task, which consists of two fully connected hidden layers with dimensions of 256 and 128 neurons respectively. The model is optimized using the Adam optimizer with a learning rate of 1e-3

- **FMNIST[24]** is a variant of MNIST dataset, consisting of Zalando's clothing images, with 60,000 training images and 10,000 test images of the same size as MNIST. The images are also 28x28 pixel greyscale maps, but the categories are more complex, covering 10 different clothing styles. The model used for FMNIST is a Convolutional Neural Network (CNN) with two convolutional layers and two fully connected layers. The first convolutional layer consists of 32 filters with a kernel size of 3x3, followed by a ReLU activation function. The second convolutional layer contains 64 filters of the same size, followed by ReLU and max-pooling with a kernel size of 2x2.

- **CIFAR-10[25]** dataset contains 60,000 colour images with an image size of 32x32 pixels, divided into 10 categories covering aircraft, cars, birds, cats, deer, dogs, frogs, horses, boats and trucks. Each category contains 6,000 images and the complexity of the dataset is reflected in its RGB structure. In particular, the CIFAR10 dataset is divided into two different variants in order to compare the impact of data augmentation on model performance:

- **CIFAR-10 without Data Augmentation.** In this version, the training set is only treated with basic image normalization. The image pixel values are normalized to keep the original image content unchanged.
- **CIFAR-10 with Data Augmentation.** This version of the training set applies data augmentation techniques, including random cropping and horizontal flipping, to increase the diversity of the data with the aim of improving the generalization ability of the model. Specific augmentation operations include random cropping (with 4-pixel padding) on a 32 x 32 pixel image, and random horizontal flipping[26].

The use of two different treatments of the CIFAR-10 dataset is a common practice when evaluating inference attacks. The logic behind this lies in artificially creating a situation of high model overfitting levels by deliberately not applying data augmentation techniques to complex datasets, thus facilitating the assessment of the extreme performance of inference attacks. The model used for both data versions is a simplified MobileNet, using depthwise separable convolutions to reduce the number of parameters. Its architecture includes a series of convolutional layers followed by batch normalization and ReLU activations and a fully connected layer for 10 classes classification task.

6.1.2 Topology Setting

The DFL network involved in all the experiments consists of the following five topologies:

- **Ring Graph:** A ring topology consists of network nodes connected in a closed loop along a fixed direction, with each node connected to its left and right neighboring nodes in a point-to-point closed structure. This topology allows each node to have the same degree and makes the communication inside follow a circular path.
- **Star Graph:** In a star topology, the nodes of a network are connected to a central node in a point-to-point manner. This topology is a typical asymmetric structure.
- **ER Graph (p=0.3):** ER graph represents Erdős-Rényi graph[27], which is a famous type of random graph. In this topology, a fixed number of nodes are connected by edges based on a probability p. Each edge's probability is independent from every other edge. Here the probability is set as 0.3.
- **ER Graph (p=0.5):** Erdős-Rényi random graph with the edge probability setting as 0.5.
- **ER Graph (p=0.7):** Erdős-Rényi random graph with the edge probability setting as 0.7.

These five evaluated topologies cover three main types of topologies: cyclic graphs, asymmetric graphs, and random graphs, and also range from sparse graphs to dense graphs. Therefore, these topologies can provide a comprehensive evaluation basis for attacks and help analyze performance under different network structures.

6.1.3 Federated Learning Configuration

The configuration of Decentralized Federated Learning involoved in all experiments are summarized as follows:

- **Number of Nodes:** The total number of nodes in DFL is set to 10, 20, and 30, representing small, medium, and large networks respectively. However, the total number of training datasets involved in the whole federated learning is fixed at 25,000 for any network size, and this setting also applies to different types of datasets.

  In other words, the size of the training dataset held by each node is 2500, 1250, and 834 for networks of size 10, 20, and 30, respectively. This method of data partitioning helps control the impact of data size on the level of overfitting of the model in the federation.

- **Total Rounds:** Considering the different efficiencies of node model information diffusion in networks of different sizes, the total number of DFL rounds is set to 10, 30, and 40 for networks with 10, 20, and 30 nodes, respectively.

- **Local Epochs:** For networks of different sizes, the local epochs are set to 3 and 10 in two separate scenarios. By setting different local epochs, it is possible to study the impact of different levels of local overfitting on the effectiveness of the attack.

- **Data Distribution:** The training dataset between each node is always set to comply with the independent and identical distribution hypothesis.

6.1.4 Attack Models

The attack models involved in the experiments of topology inference attacks are divided into two categories according to different attack scenarios:

1. **Attack Scenario 1 (Full Node Information and Partial Edge Information):** As described in Section 5.3, the attack model in this attack scenario belongs to the type of supervised machine learning model. Therefore, the following types of models are tested separately for their effectiveness:

   - **Logistic Regression[28]:** Logistic regression is a linear classification model. It can predict the connections between nodes by learning the relationship between node metrics and edge information. Despite its simple model, it has robust performance when dealing with binary classification problems. It also trains faster and is more suitable for dealing with sparse graph structures.
   - **Support Vector Machine (SVM)[29]:** SVM is a widely used classification model. It distinguishes between connected and unconnected nodes by searching for an optimal hyperplane in high-dimensional space. SVM excels at handling complex, non-linear boundaries, especially when the data is high-dimensional.

- **Random Forest (RF)[30]:** Random forest is an ensemble learning method based on decision trees. It improves the generalization ability of the model by constructing multiple decision trees and combining their prediction results. Random forest can effectively handle non-linear relationships in node metrics and are robust in the face of data noise.

2. **Attack Scenario 2 (Only Full Node Information):** Similarly, the attack model used in this attack scenario belongs to the type of clustering models in the category of unsupervised machine learning. The following clustering models are selected for evaluation:

    - **K-Means[25]:** K-Means is a distance-based clustering algorithm. It infers the connection between nodes by dividing the node features into multiple clusters. It then ensures that nodes within the same cluster are more similar by minimizing the variance of the node features within the cluster. It is often used to process large data sets because of its advantage of fast calculation.
    - **Gaussian Mixture Model (GMM)[31]:** GMM is a probabilistic clustering algorithm. It first assumes that the data consists of multiple Gaussian distributions, and then infers the probability that the nodes belong to different clusters by maximizing the likelihood estimation. Compared to K-Means, GMM can handle complex data distributions more flexibly. At the same time, it is also suitable for capturing the potential probabilistic relationships between features.
    - **Spectral Clustering[32]:** Spectral clustering is an algorithm that uses the Laplacian matrix of a graph to perform eigenvalue decomposition of the similarity matrix between nodes, and then maps the nodes to a low-dimensional space before clustering. It often performs better when processing data based on a graph structure.

The attack models for the different attack scenarios above are all combined with the different data processing algorithms mentioned in Section 5.3 during evaluation to determine which method poses the greatest attack threat.

6.1.5 Evaluation Metric

For either attack scenario, the designed attack strategy is to obtain a complete or partial inferred adjacency matrix that is as plausible as possible, so in order to evaluate the final performance of attacks, it is essentially a matter of comparing the similarity of the inferred adjacency matrix to the original one. This transforms this issue into evaluating a binary classification problem. Therefore, it is a reasonable choice to adopt the F1-Score as the final attack performance evaluation metric.

The F1-Score is the harmonic mean of classification precision and recall[33], providing a single metric that balances both concerns. The F1-Score is defined as:

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{6.1}$$

Attack Scenario	**Dataset**	**Topology**	**Num of Nodes**	**Threat Model**	**Total Rounds**	**Evaluated Metric**
Scenario 1	MNIST, FMNIST, CIFAR10no, CIFAR10	Ring, Star, ER_0.3, ER_0.5, ER_0.7	10, 20, 30	LR, SVM, RF	10, 30, 40	F1-Score
Scenario 2				Kmeans, GMM, Spectral		

Table 6.1: Overall Experiment Setup.

A higher F1-Score indicates a better balance between precision and recall, making it a comprehensive metric for overall attack performance.

In summary, all the experimental details available for evaluating topology inference attacks are summarized in the Table6.1.

6.2 Experiment Results

This section presents the evaluation results of the performance of topology inference attacks in two attack scenarios in different environments. First, the impact of different data processing methods and node metrics on the effectiveness of the attack is shown. The results are compared to determine the best data processing method when using the clustering model for attacks and the most effective node metrics in both attack scenarios. Second, the performance of topology inference attacks is evaluated for different datasets and network topologies. Finally, the difference of the attack performance is analyzed for different number of DFL local training epochs, number of nodes and number of global rounds. Based on this comprehensive experimental results, it helps to analyze in depth the endogenous and exogenous factors affecting the effectiveness of topology inference attacks in DFL network.

6.2.1 Node Metrics and Data Processing Method Selection

Figure6.1 to 6.2 show the impact of the three different data processing methods on the attack performance in the second attack scenario, where a clustering model is used as the attack model. Here, all cases use the Relative Loss node metric as the guidance of attack. From these three figures, it is clear to see the impact of these three data processing methods on the attack effectiveness. Among them, the results obtained by Algo1 are significantly worse than the other two methods, especially in the context of the star graph. Algo2 and Algo3 perform more or less equally. Therefore, it is feasible to choose either of these two as the data processing method for the next evaluation of the performance of the cluster attack. Algo3 is selected here because it classifies by constructing a two-dimensional data group, which can show more data relationships.

The reason for the poor results of Algo1 is also relatively easy to identify.The reason is that it performs clustering of objects by constantly analyzing each row of the node metric matrix to obtain the final inference result. Although this approach to data processing reduces the complexity of running clustering algorithms on large network datasets, it is very susceptible to biased estimation because it considers fewer samples each time (only 1/|V| compared to the other two approaches).

As for the effects of different node metrics on the attack performance, they are clearly shown in Figure6.4 to Figure6.6. The performance of the results in these plots shows that the attack constructed via Relative Loss performs the best, with Euclidean Distance and Relative Entropy next in line, and the Cosine Similarity metric being the worst. Such attack results are in line with the previous metric validation results, where the distribution distinction between the edge group and non-edge group for the Relative Loss was the largest in the validation results above, thus indicating the most potential to perform threatening attacks. The cosine similarity, on the other hand, is too high due to the overlap between the two groups, indicating that the cosine similarity between each node model is very close, making it less suitable as a successful node metric for topology inference. Here, the evaluation results also corresponds to this argument.

6.2.2 Performance Comparison on Dataset and Topology Aspects

After determining the optimal data processing method and node metrics, the performance of the attack can be analyzed under different datasets and topologies. This result is clearly presented in Figure6.3. Firstly, analyzing under the perspective of the dataset, it can be found that the CIFAR10-no dataset presents the best attack, both for which clustering method. The absolute value level of the overall F1-Score is basically close to 1, indicating that the topology inference attack is excellent for this approach. Next, the attack performance on the CIFAR10 dataset stays second overall while the MNIST dataset is the least effective. This trend is also similar to the performance of other kinds of inference attacks. As mentioned earlier, an important mechanism for the success of inference attacks lies in the level of overfitting of the machine learning model. Inference attacks work better on relatively complex datasets such as CIFAR-10, and worse on datasets such as MNIST.

As for the differentiation of the effectiveness of the attack between different topology levels, its manifestation can be shown as a decrease in the effectiveness of the attack when the overall density of the network rises. For example, the attack shows very good performance in both like star and ring graphs, while the effect suffers a reduction in ER graphs with high edge connection probability. This performance can be understood as when the density of the overall network rises, the average distance of the network gets decreased and thus the frequency of information exchange between individual nodes gets elevated. As a result, the generalization ability and similarity of the models between the nodes become better, which makes the topology inference attack based on this mechanism performance reduced.

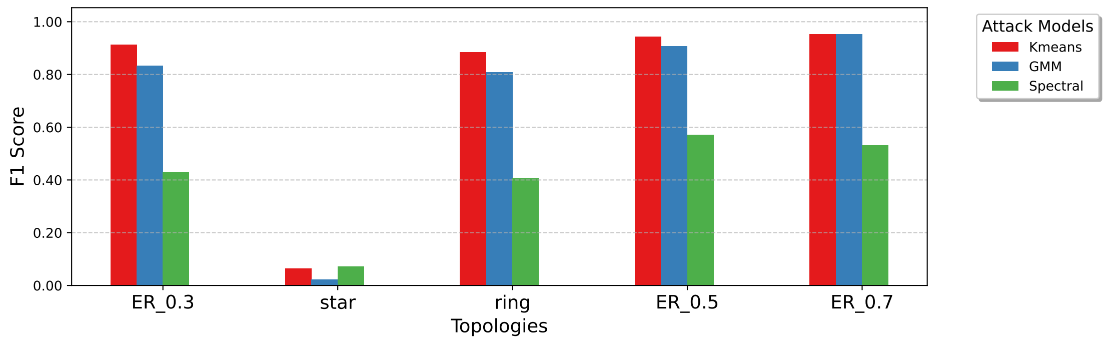
6.2.3 Performance Comparison on DFL Configuration Aspects

Different DFL parameter settings also have a large impact on the performance of topology inference attacks. The three main elements considered here are local epochs, number of nodes and number of rounds.

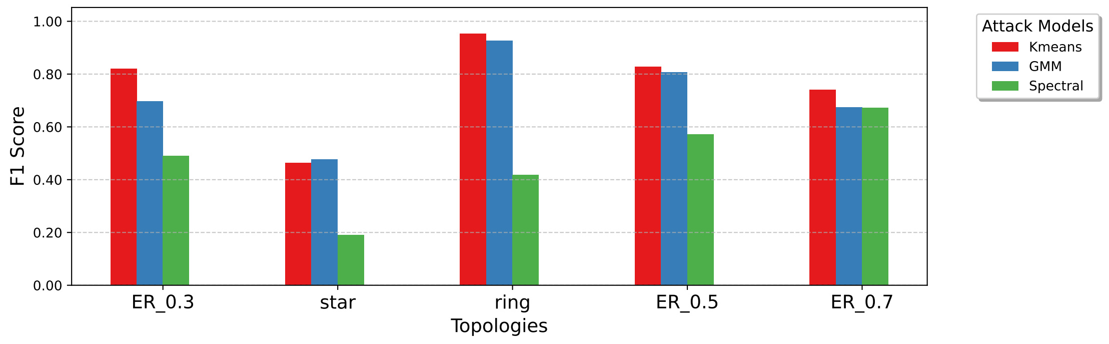
First, Figures6.3 and Figure6.7 show the difference in the performance of the attack with local epoch set to 3 and 10. Here it is clear that the performance of the attack with local epochs of 10 is much better than the case with epochs of 3. The reason for this difference should come from the fact that when the number of local epochs increases, the overfitting level of the model increases accordingly, leading to better attack performance.

Secondly, Figure6.3, Figure6.8 and Figure6.10 show the difference in the effectiveness of the attack with different number of nodes. It can be clearly seen that when the overall number of nodes in the network rises, the overall trend of attack effectiveness is decreasing. The reason for this phenomenon comes from two aspects: (1) With the expansion of the network size, the difficulty of topology inference itself has been greatly improved, and the decline in attack performance is also in accordance with logic; (2) When the number of nodes in the DFL network continues to rise, the density and average distance of the overall network are also greatly improved, and this change in topology makes the model between the nodes become more similar, thus making the inference attack perform poorly.

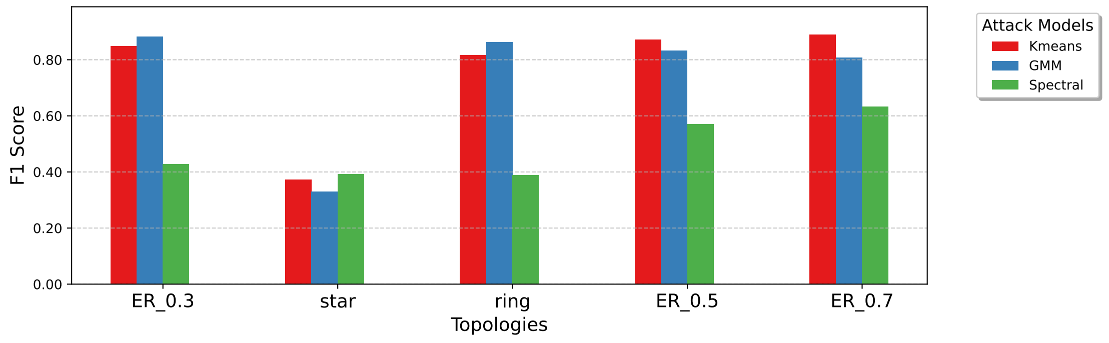
Last, Figure6.9 and Figure6.8 show the difference in the performance of the attack under round 9 and round 29 in a DFL network with 20 nodes. Here it can be seen that the overall level of the two attacks does not change much, which suggests that in DFL networks, the individual node models tend to propagate faster than the preset rounds, which may allow the attacks to get better within the earlier rounds of the overall run.



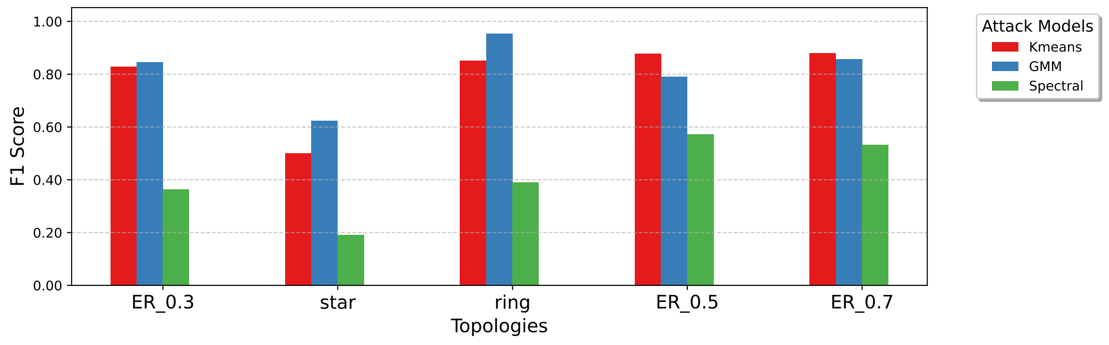
(a) CIFAR-10no



(b) CIFAR-10

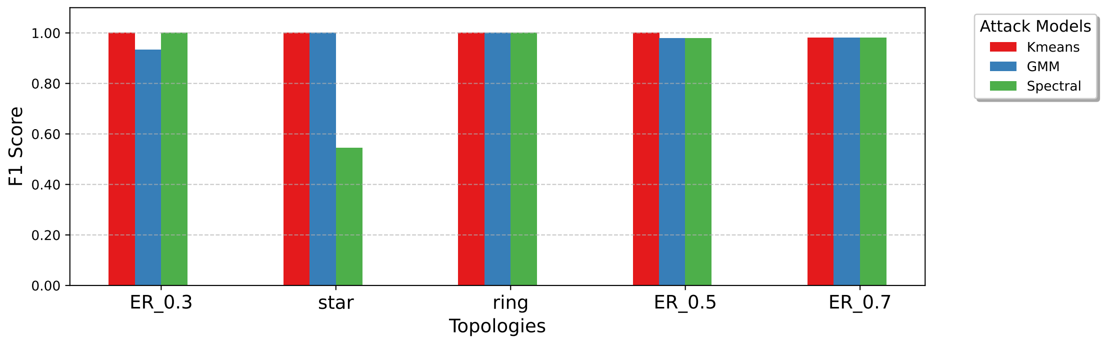


(c) MNIST

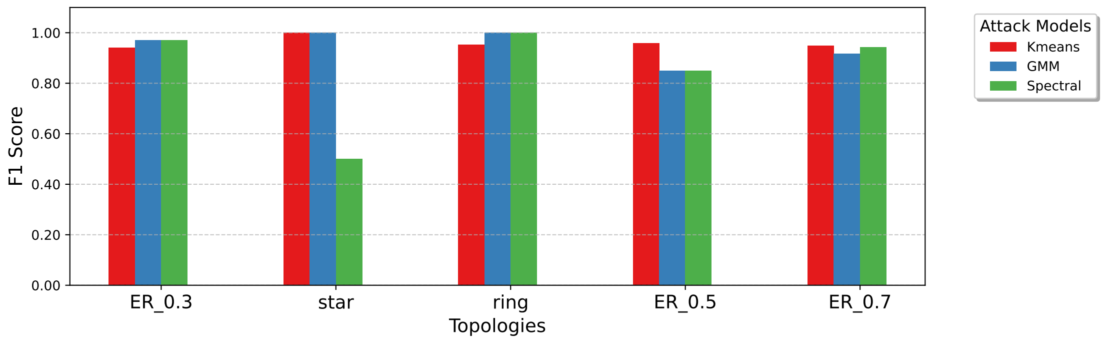


(d) FMNIST

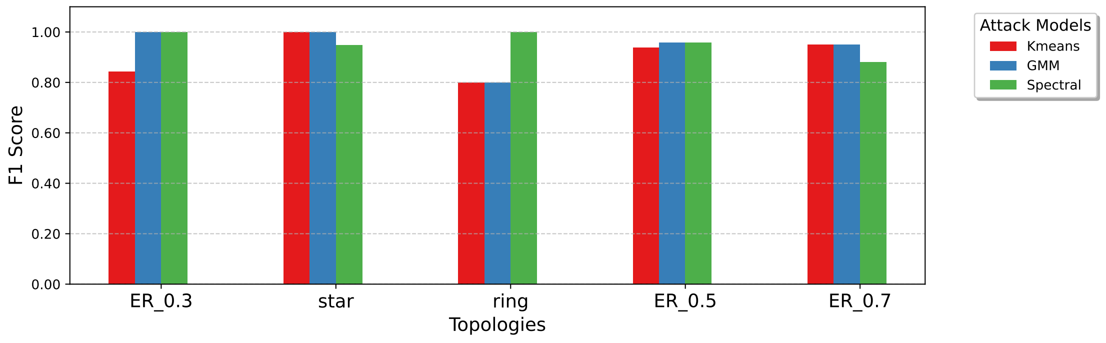
Figure 6.1: Attack Performance based on Relative Loss calculated from Algo1 under 10 Nodes DFL Network with 10 local epochs.



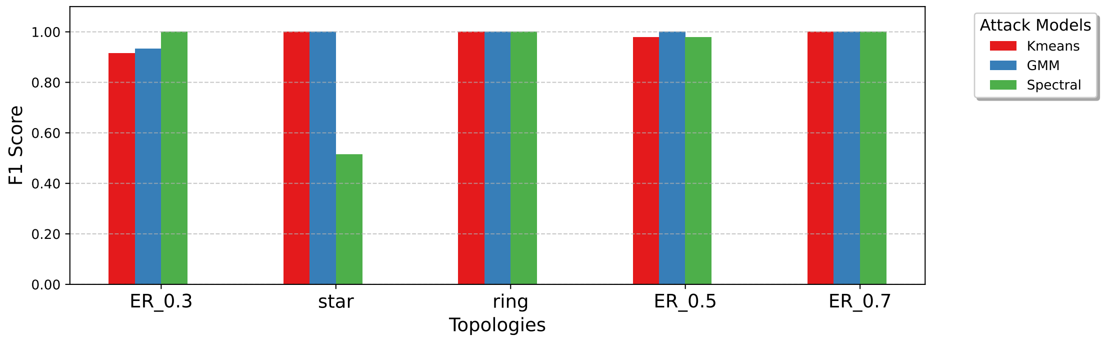
(a) CIFAR-10no



(b) CIFAR-10

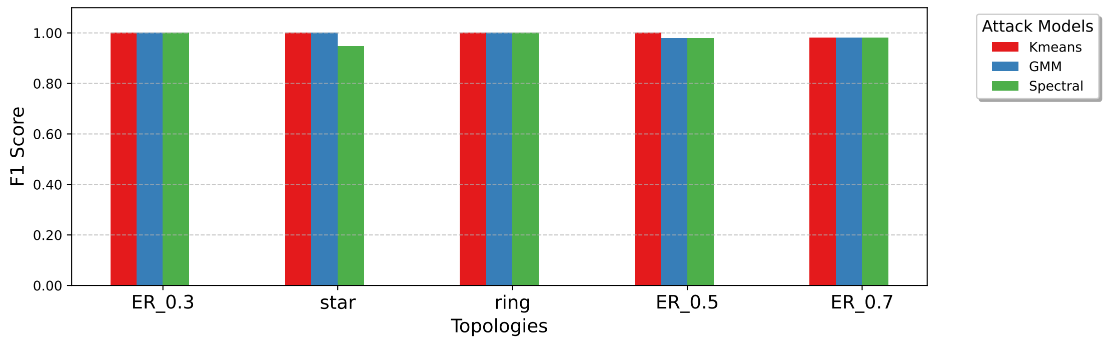


(c) MNIST

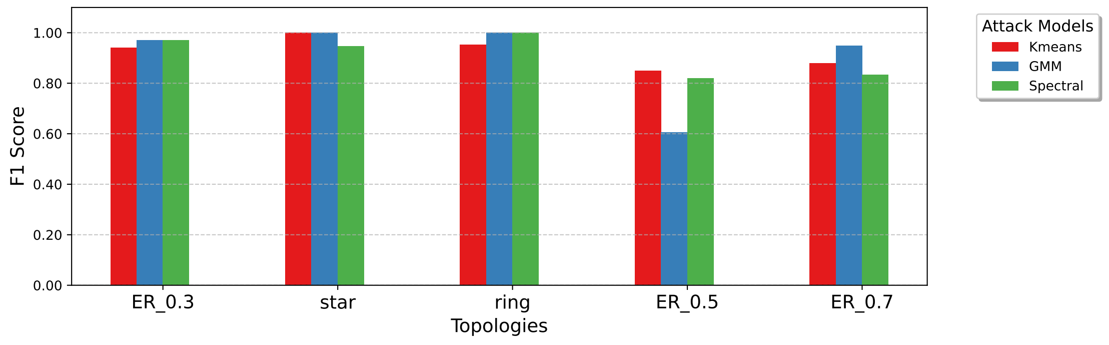


(d) FMNIST

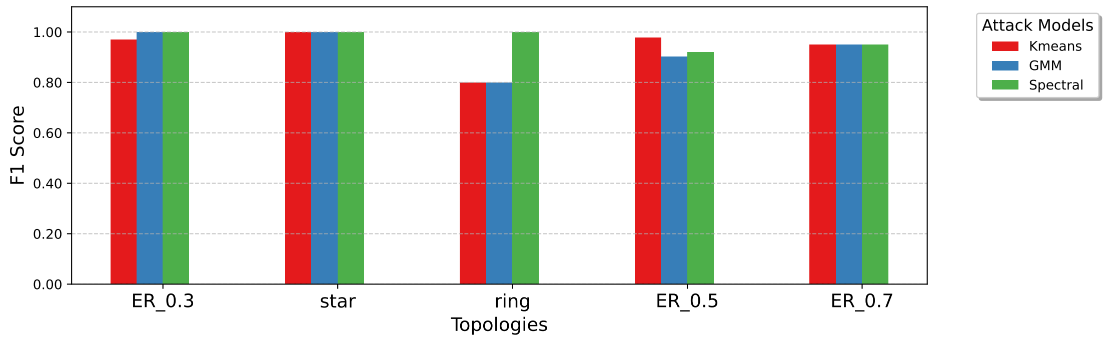
Figure 6.2: Attack Performance based on Relative Loss calculated from Algo2 under 10 Nodes DFL Network with 10 local epochs.



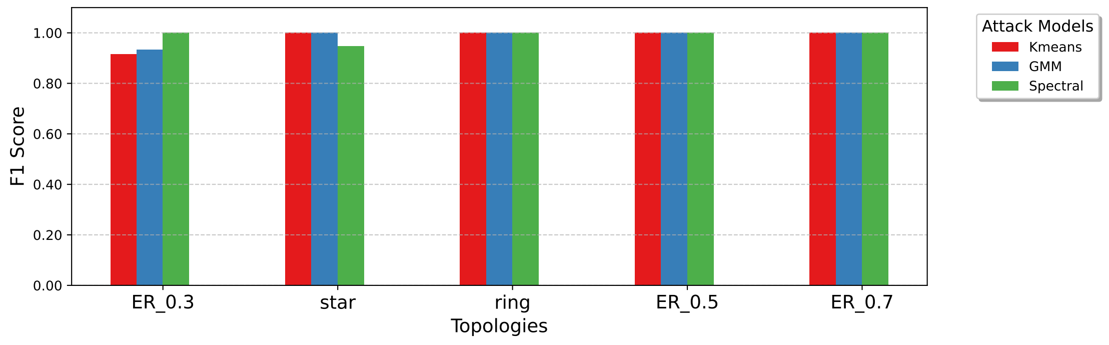
(a) CIFAR-10no



(b) CIFAR-10

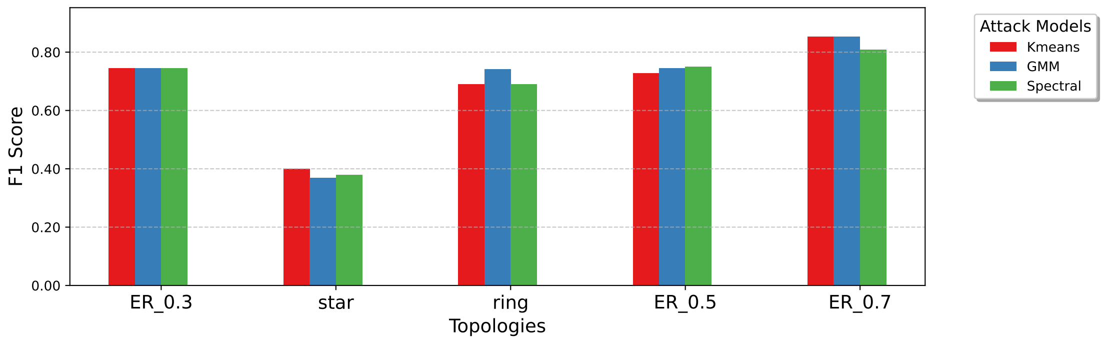


(c) MNIST

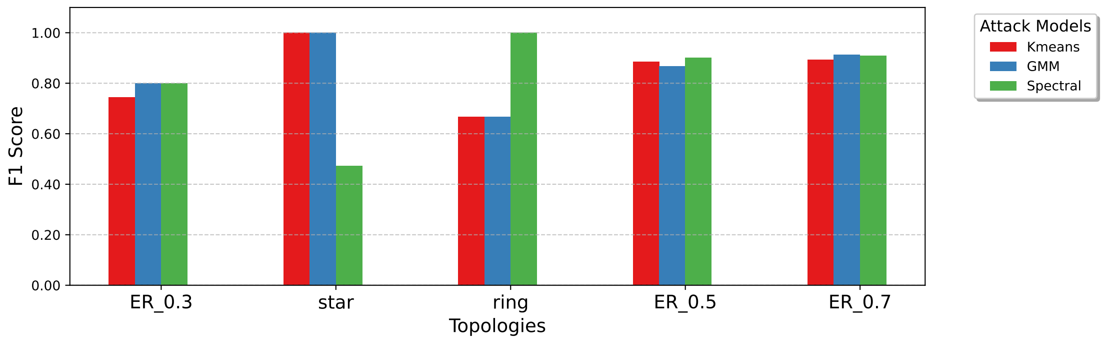


(d) FMNIST

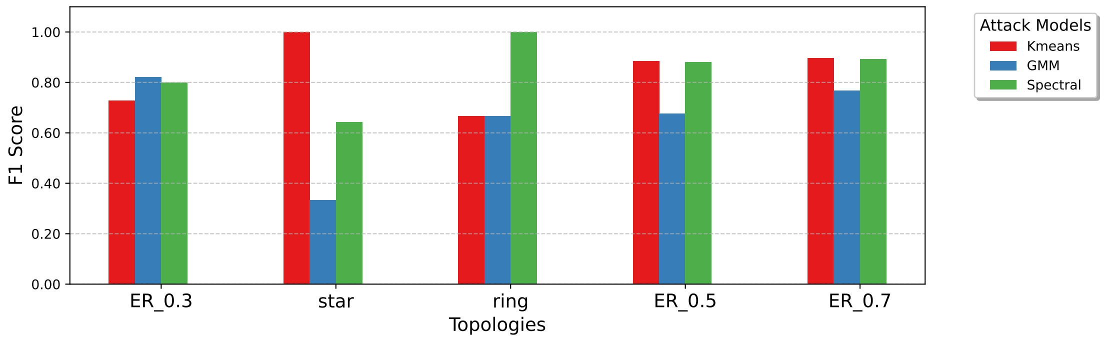
Figure 6.3: Attack Performance based on Relative Loss calculated from Algo3 under 10 Nodes DFL Network with 10 local epochs.



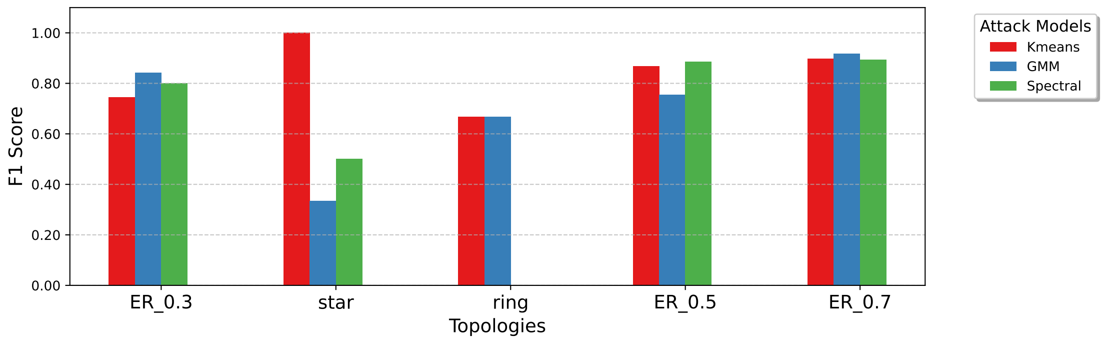
(a) CIFAR-10no



(b) CIFAR-10

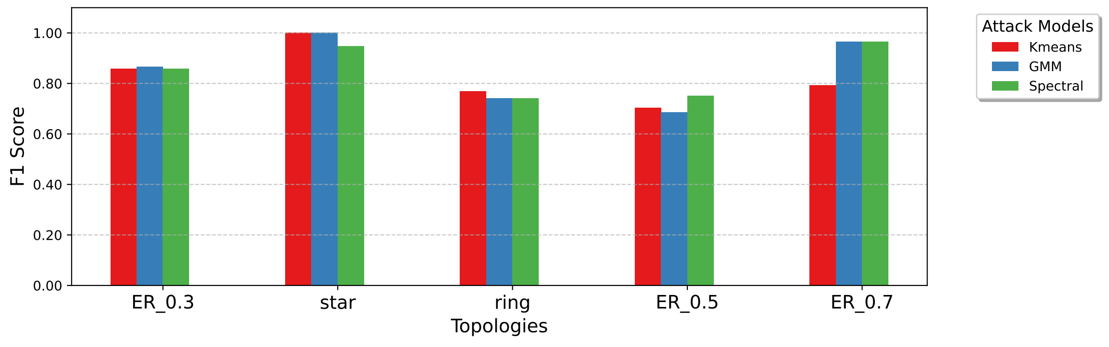


(c) MNIST

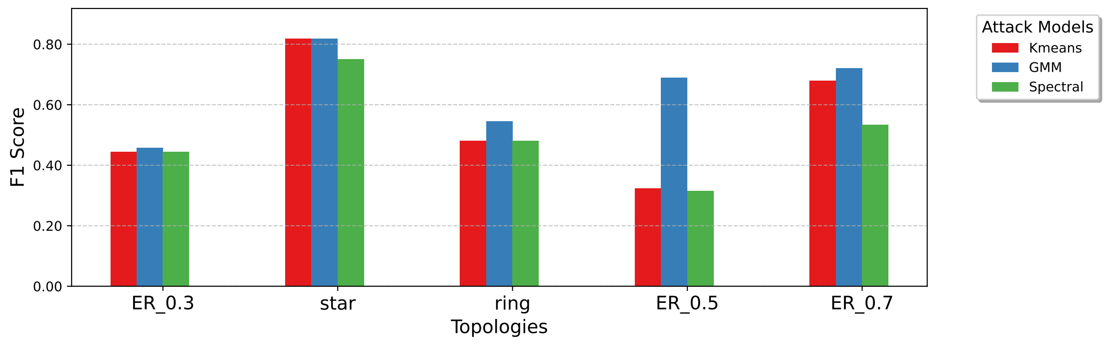


(d) FMNIST

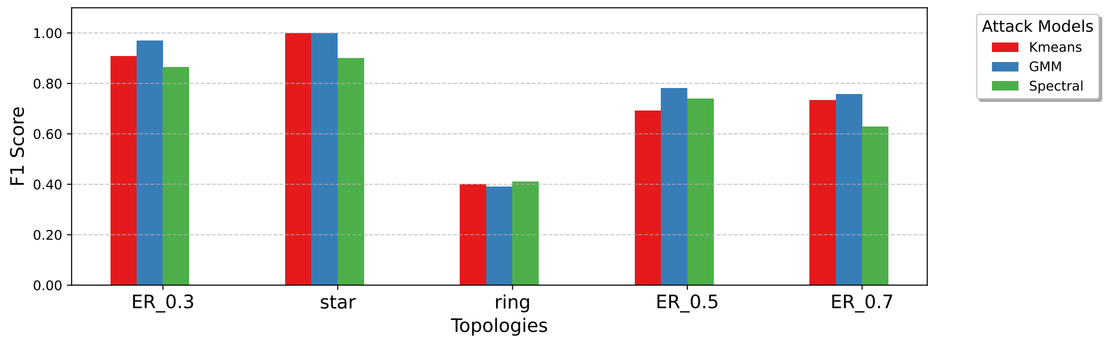
Figure 6.4: Attack Performance based on Cosine Similarity under 10 Nodes DFL Network with 10 local epochs.



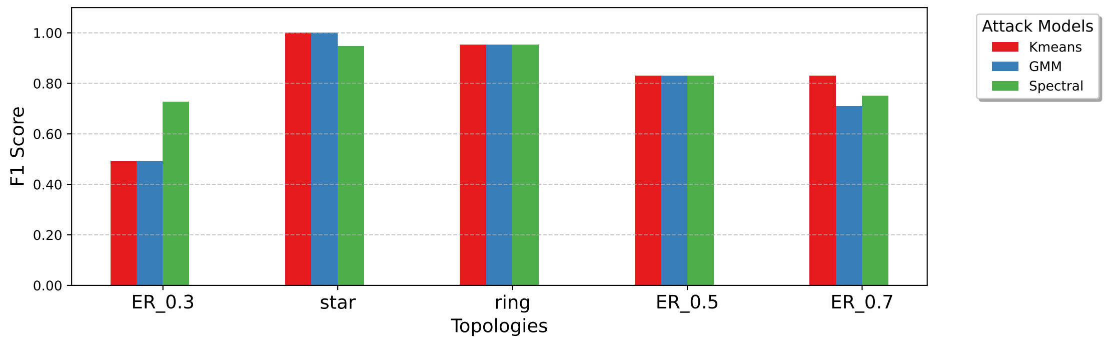
(a) CIFAR-10no



(b) CIFAR-10

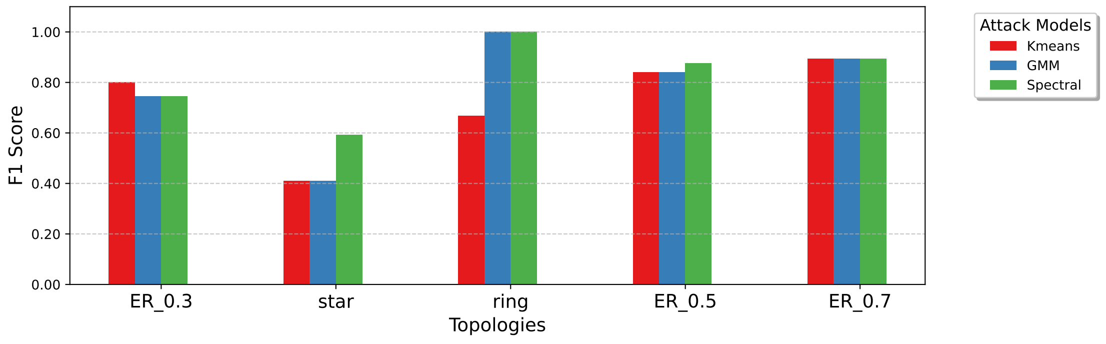


(c) MNIST

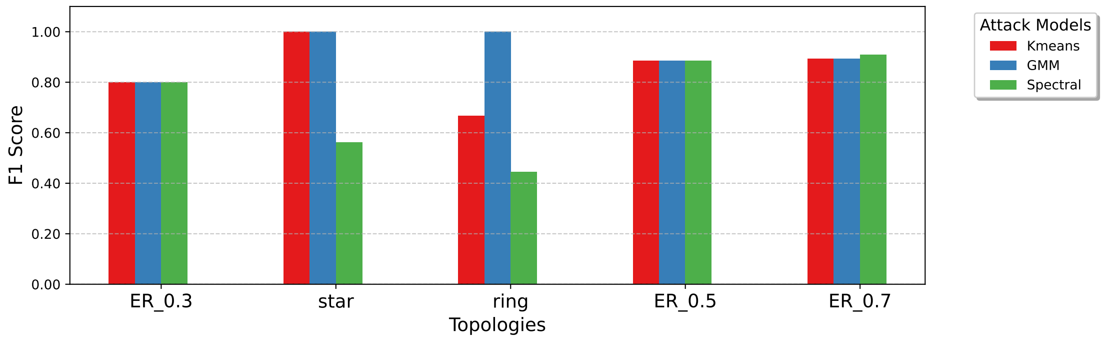


(d) FMNIST

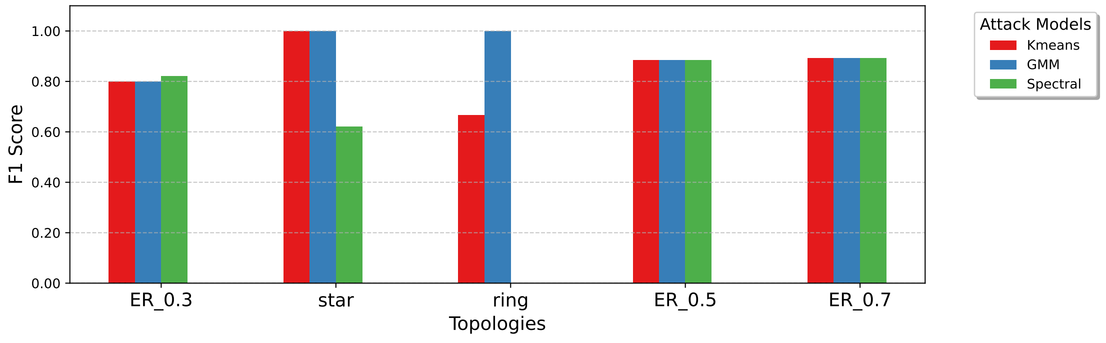
Figure 6.5: Attack Performance based on Relative Entropy under 10 Nodes DFL Network with 10 local epochs.



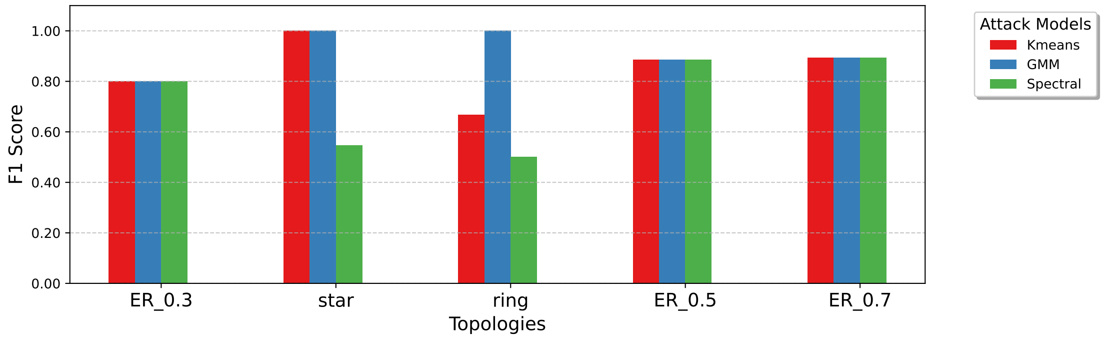
(a) CIFAR-10no



(b) CIFAR-10

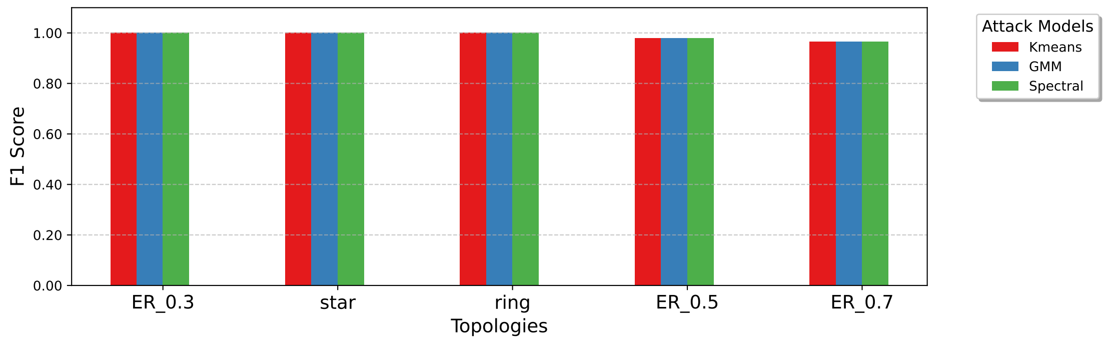


(c) MNIST

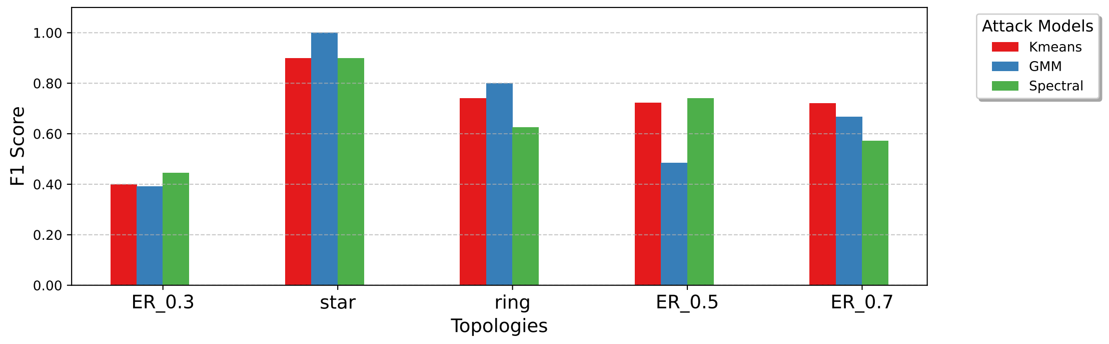


(d) FMNIST

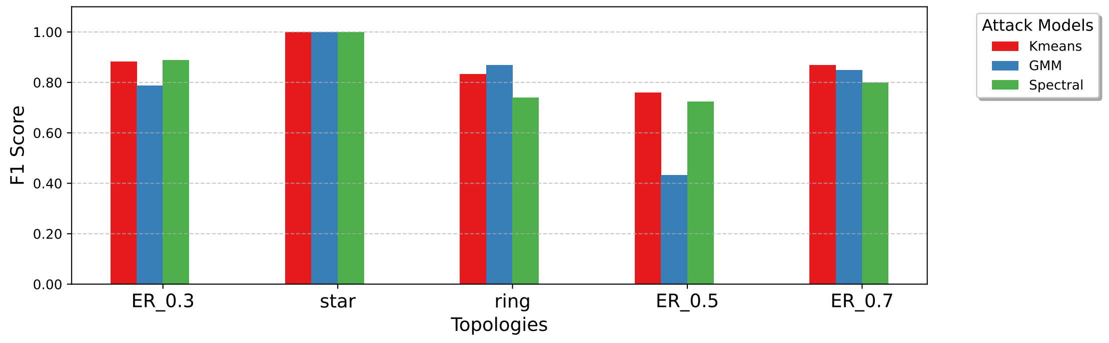
Figure 6.6: Attack Performance based on Euclidean Distance under 10 Nodes DFL Network with 10 local epochs.



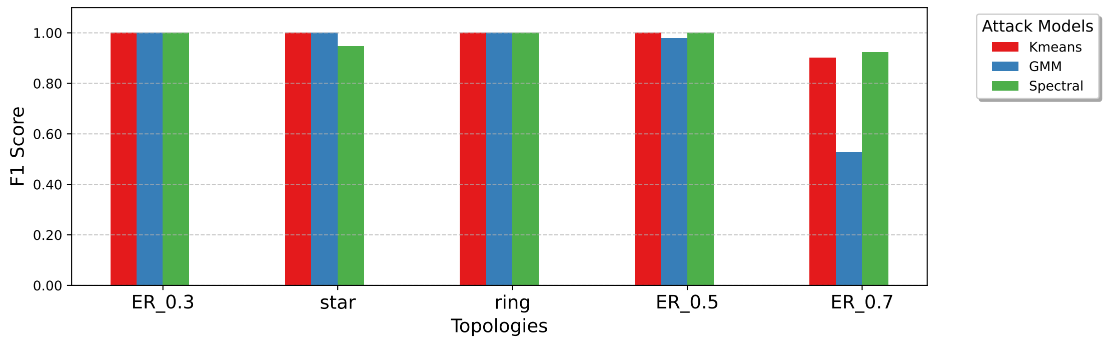
(a) CIFAR-10no



(b) CIFAR-10

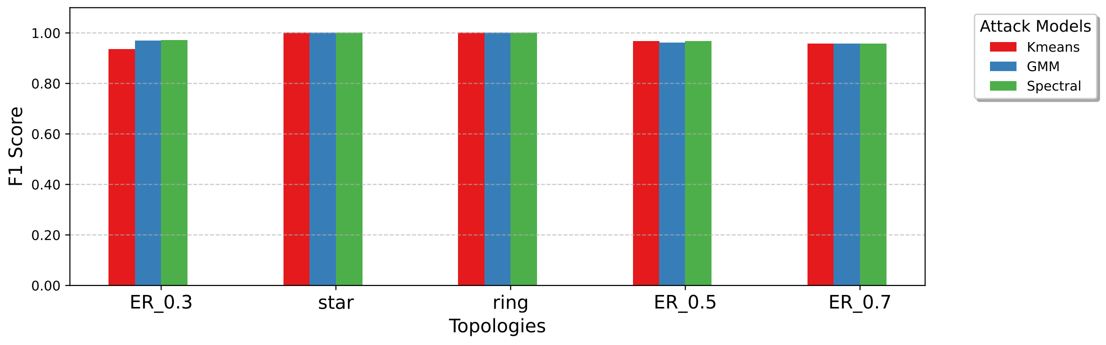


(c) MNIST

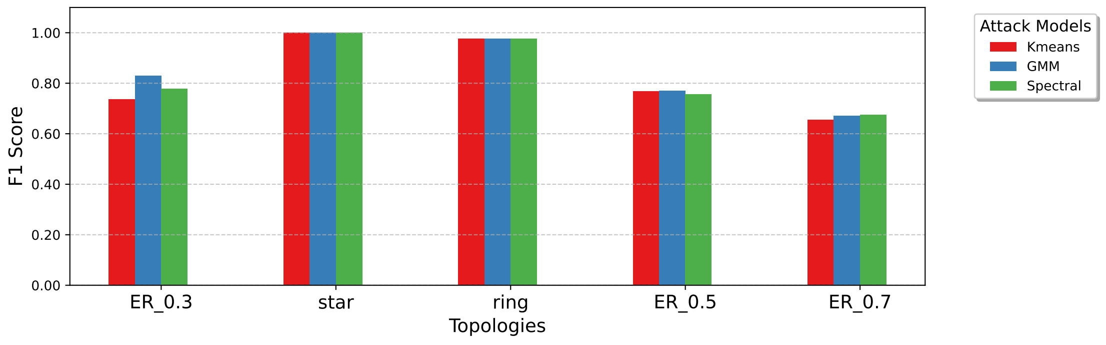


(d) FMNIST

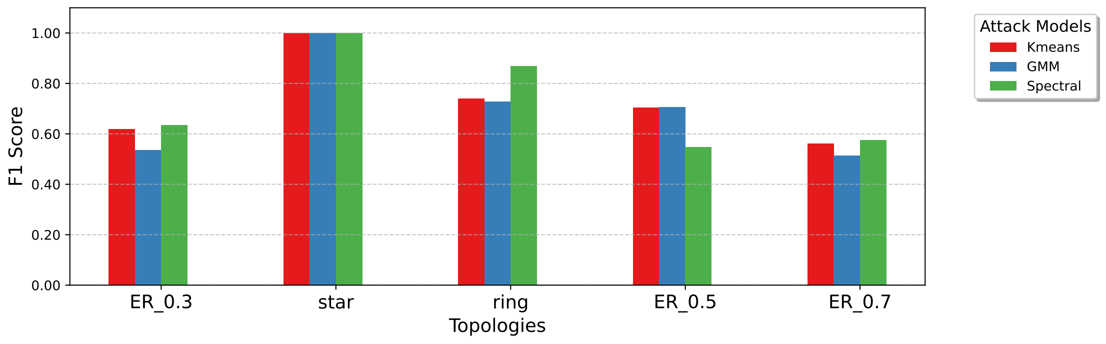
Figure 6.7: Attack Performance based on Relative Loss under 10 Nodes DFL Network with 3 local epochs.



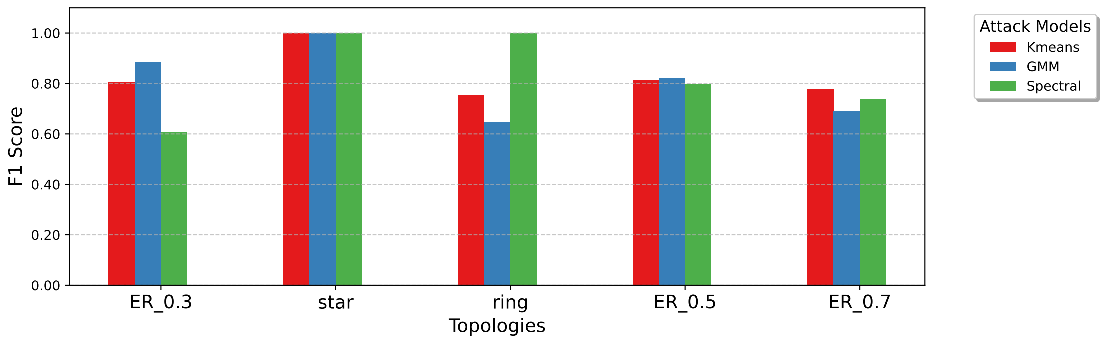
(a) CIFAR-10no



(b) CIFAR-10

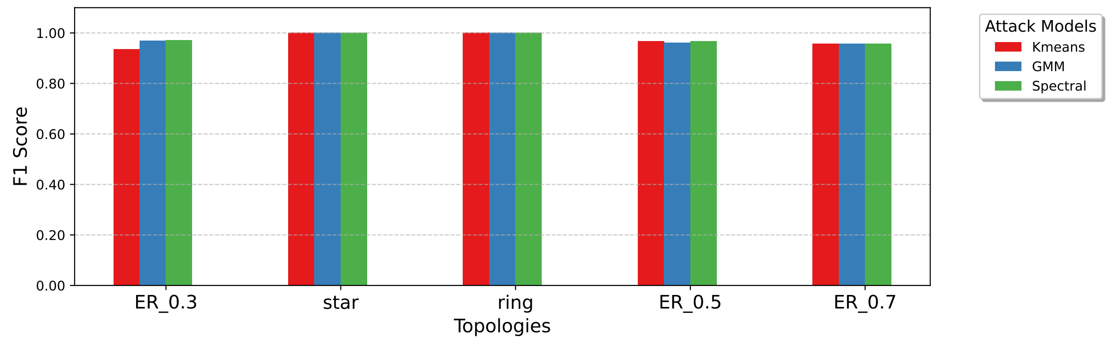


(c) MNIST

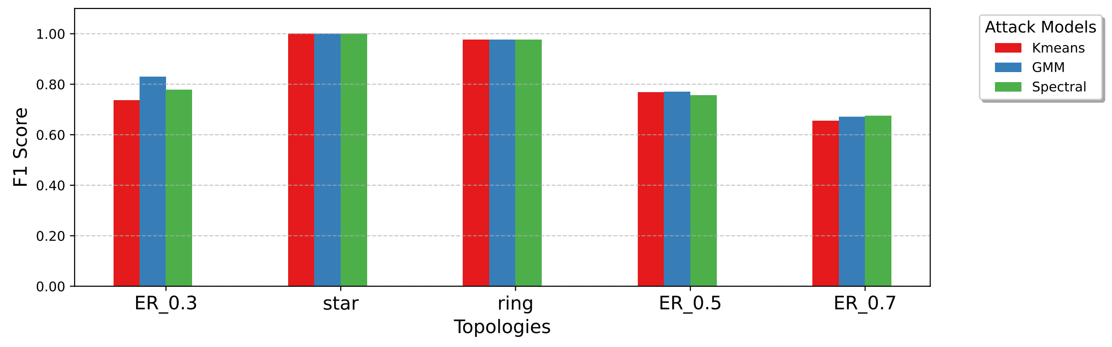


(d) FMNIST

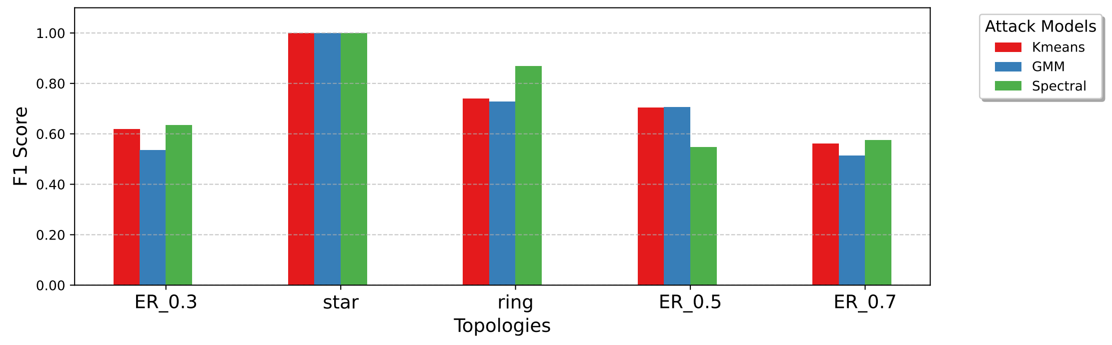
Figure 6.8: Attack Performance based on Relative Loss under 20 Nodes DFL Network with 10 local epochs and 29 Global Rounds.



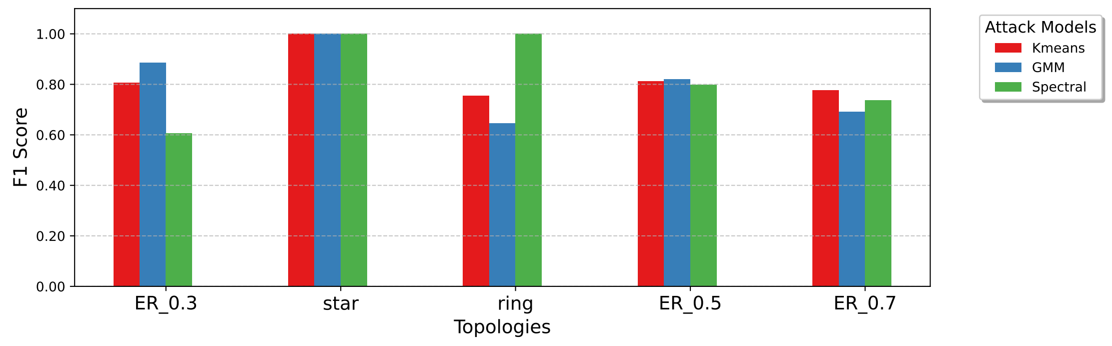
(a) CIFAR-10no



(b) CIFAR-10

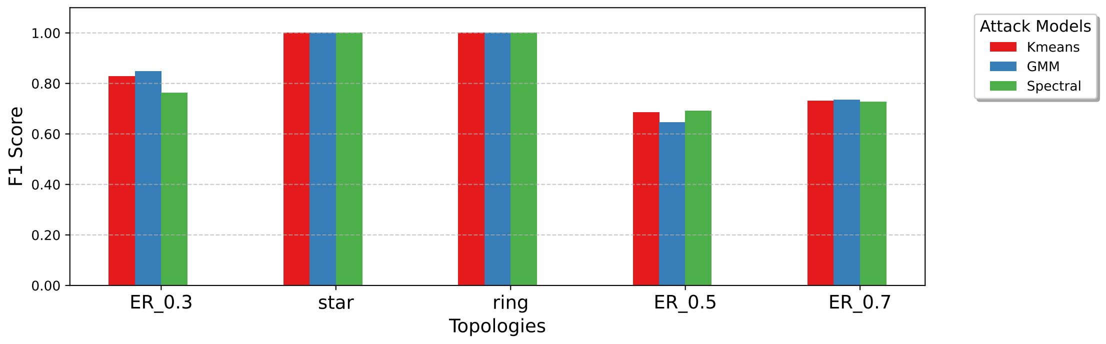


(c) MNIST

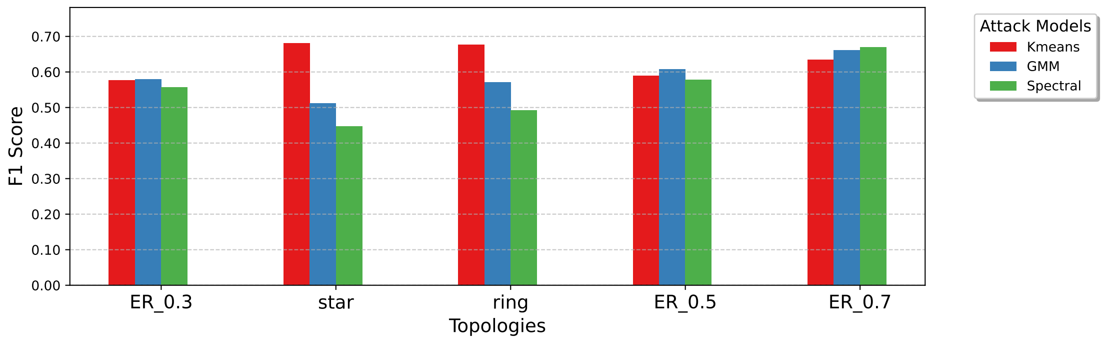


(d) FMNIST

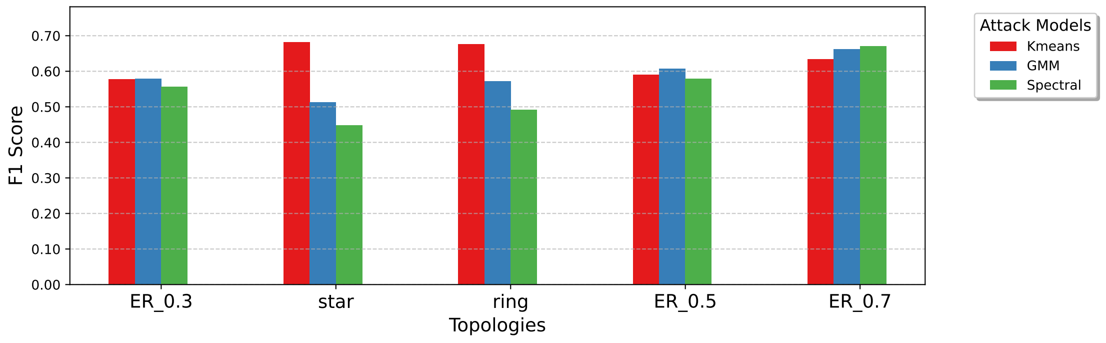
Figure 6.9: Attack Performance based on Relative Loss under 20 Nodes DFL Network with 10 local epochs and 9 Global Rounds.



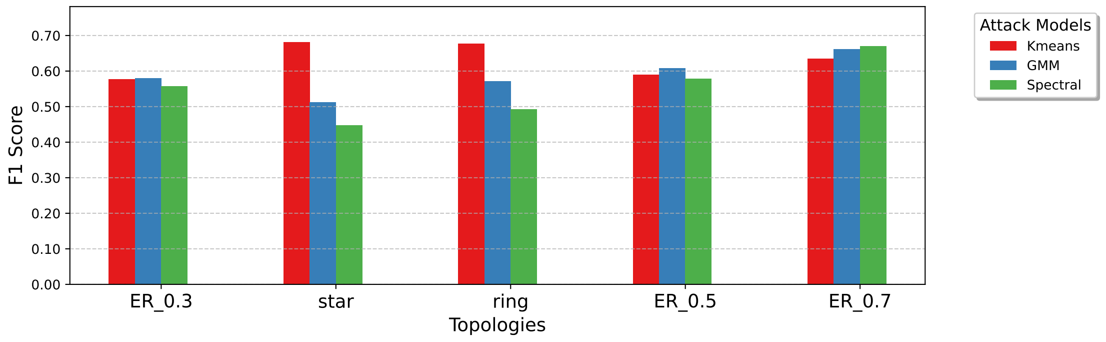
(a) CIFAR-10no



(b) CIFAR-10



(c) MNIST



(d) FMNIST

Figure 6.10: Attack Performance based on Relative Loss under 30 Nodes DFL Network with 10 local epochs and 39 Global Rounds.

Chapter 7

Summary and Conclusions

This work focuses on designing a topology inference attack specifically for DFL networks. This attack obtains the topology of a DFL network by innovatively utilizing information about the nodes suitable for that DFL network. Since this type of attack has never been proposed before in the field of DFL research, the first step is to learn similar methods from other well-known network fields such as computer networks and social networks and extend them to the DFL area. Meanwhile, in order to better analyze the different scenarios of attacks in DFL networks, two different attack scenarios are hypothesized based on the status of the information the attacker has about the nodes.

The first attack scenario assumes that the attacker has information about all the nodes as well as some of the edges. In this case, the attacker can try to construct links from the existing edge information and the corresponding node information by training a supervised machine learning model, and predict the unknown edge connectivity through such model. This approach is justified by the fact that the correspondence between edges and nodes applies to all individuals under the same network. Therefore generalizing to unknown predictions by capturing features from known parts is a more reasonable means. The second attack scenario assumes that the attacker only has information about all the nodes and does not know any information about the edges. In this scenario, an approach that utilizes clustering models under unsupervised learning is introduced. Based on all the node metrics, the clustering model is used to classify the nodes into distinct two categories-edge group and non-edge group, so that the connectivity between pairs of nodes can be constructed. This method of attack is similar to autonomously demarcating boundaries from an existing dataset. However, since the number of delineated categories is fixed and clear, two nodes are either directly connected or not, making the overall approach highly generalizable.

Although the type and level of knowledge that can be utilized by an attacker differs in the two different attack scenarios, the node metric is a required element in either case. Therefore, this work proposes and constructs four valid node metrics suitable for use in DFL networks-Relative Loss, Relative Entropy, Relative Entropy, and Euclidean Distance. The first two of these address the generizability between node local models, while the latter two address the similarity of the model between nodes. This study then presents a comprehensive evaluation of attack methods across various scenarios, driven by four different node metrics in diverse DFL network settings.

The evaluation results regarding the designed attack can be summarized as follows:(1) The impact of different datasets on the effectiveness of the attack is large. The topology inference attack tends to achieve better performance when facing more complex datasets such as CIFAR-10, etc.; (2) The topology of the DFL network also deeply affects the effectiveness of the attack. This effect is manifested in two main ways: when the network size increases, the attack performance decreases significantly as well as when the network becomes denser. The reason for this phenomenon can be attributed to the change in the level of overfitting of the overall network model; (3) Among the alternative four node metrics, Relative Loss achieve the best performance level. This suggests that even in a DFL network that has been trained for many rounds, the mutual generizability between the nodes models can still perform poorly, thus providing a window for some attacks to target.

Overall, the topology inference attack designed in this work for DFL networks has achieved good results in most of the DFL networks, and at the same time, it serves as a very innovative attack and will provide a new way of thinking for research in the field of DFL network security afterwards.

Bibliography

[1] E. T. M. Beltrán, Á. L. P. Gómez, C. Feng, *et al.*, "Fedstellar: A platform for decentralized federated learning", *Expert Systems with Applications*, vol. 242, p. 122 861, 2024.

[2] X. Yin, Y. Zhu, and J. Hu, "A comprehensive survey of privacy-preserving federated learning: A taxonomy, review, and future directions", *ACM Computing Surveys (CSUR)*, vol. 54, no. 6, pp. 1–36, 2021.

[3] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models", in *2017 IEEE Symposium on Security and Privacy (SP)*, 2017, pp. 3–18. DOI: `10.1109/SP.2017.41`.

[4] A. Salem, Y. Zhang, M. Humbert, P. Berrang, M. Fritz, and M. Backes, "Ml-leaks: Model and data independent membership inference attacks and defenses on machine learning models", *arXiv preprint arXiv:1806.01246*, 2018.

[5] S. Dayal, D. Alhadidi, A. Abbasi Tadi, and N. Mohammed, "Comparative analysis of membership inference attacks in federated learning", in *Proceedings of the 27th International Database Engineered Applications Symposium*, 2023, pp. 185–192.

[6] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures", in *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, 2015, pp. 1322–1333.

[7] Z. He, T. Zhang, and R. B. Lee, "Model inversion attacks against collaborative inference", in *Proceedings of the 35th Annual Computer Security Applications Conference*, 2019, pp. 148–162.

[8] N. Z. Gong and B. Liu, "Attribute inference attacks in online social networks", *ACM Transactions on Privacy and Security (TOPS)*, vol. 21, no. 1, pp. 1–30, 2018.

[9] H. Hu, Z. Salcic, L. Sun, G. Dobbie, P. S. Yu, and X. Zhang, "Membership inference attacks on machine learning: A survey", *ACM Computing Surveys (CSUR)*, vol. 54, no. 11s, pp. 1–37, 2022.

[10] R. Motamedi, R. Rejaie, and W. Willinger, "A survey of techniques for internet topology discovery", *IEEE Communications Surveys & Tutorials*, vol. 17, no. 2, pp. 1044–1065, 2014.

[11] P. Marchetta, P. Mérindol, B. Donnet, A. Pescapé, and J.-J. Pansiot, "Topology discovery at the router level: A new hybrid tool targeting isp networks", *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 9, pp. 1776–1787, 2011.

[12] B. Donnet, P. Raoult, T. Friedman, and M. Crovella, “Deployment of an algorithm for large-scale topology discovery”, *IEEE journal on selected areas in communications*, vol. 24, no. 12, pp. 2210–2220, 2006.

[13] R. Castro, M. Coates, G. Liang, R. Nowak, and B. Yu, “Network tomography: Recent developments”, 2004.

[14] X. Jin, W.-P. K. Yiu, S.-H. G. Chan, and Y. Wang, “Network topology inference based on end-to-end measurements”, *IEEE Journal on Selected areas in Communications*, vol. 24, no. 12, pp. 2182–2195, 2006.

[15] N. Spring, R. Mahajan, D. Wetherall, and T. Anderson, “Measuring isp topologies with rocketfuel”, *IEEE/ACM Transactions on networking*, vol. 12, no. 1, pp. 2–16, 2004.

[16] M. Luckie, “Scamper: A scalable and extensible packet prober for active measurement of the internet”, in *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, 2010, pp. 239–245.

[17] J. Ni and S. Tatikonda, “Network tomography based on additive metrics”, *IEEE Transactions on Information Theory*, vol. 57, no. 12, pp. 7798–7809, 2011.

[18] G. Fei, J. Ye, S. Wen, and G. Hu, “Network topology inference using higher-order statistical characteristics of end-to-end measured delays”, *IEEE Access*, vol. 8, pp. 59 960–59 975, 2020.

[19] A. Coates, A. O. Hero III, R. Nowak, and B. Yu, “Internet tomography”, *IEEE Signal processing magazine*, vol. 19, no. 3, pp. 47–65, 2002.

[20] T. Hou, Z. Qu, T. Wang, Z. Lu, and Y. Liu, “Proto: Proactive topology obfuscation against adversarial network topology inference”, in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*, IEEE, 2020, pp. 1598–1607.

[21] F. Rahutomo, T. Kitasuka, M. Aritsugi, *et al.*, “Semantic cosine similarity”, in *The 7th international student conference on advanced science and technology ICAST*, University of Seoul South Korea, vol. 4, 2012, p. 1.

[22] P.-E. Danielsson, “Euclidean distance mapping”, *Computer Graphics and image processing*, vol. 14, no. 3, pp. 227–248, 1980.

[23] Y. LeCun and C. Cortes, *MNIST handwritten digit database*, `http://yann.lecun.com/exdb/mnist/`, Accessed: 2016-01-14, 2010.

[24] H. Xiao, K. Rasul, and R. Vollgraf, “Fashion-mnist: A novel image dataset for benchmarking machine learning algorithms”, *arXiv:1708.07747 [cs, stat]*, Sep. 2017, Accessed: 2023-07-11. DOI: `10.48550/arXiv.1708.07747`.

[25] A. Krizhevsky, G. Hinton, *et al.*, “Learning multiple layers of features from tiny images”, 2009.

[26] C. Shorten and T. M. Khoshgoftaar, “A survey on image data augmentation for deep learning”, *Journal of big data*, vol. 6, no. 1, pp. 1–48, 2019.

[27] L. Erdős, A. Knowles, H.-T. Yau, and J. Yin, “Spectral statistics of erdős-rényi graphs ii: Eigenvalue spacing and the extreme eigenvalues”, *Communications in Mathematical Physics*, vol. 314, no. 3, pp. 587–640, 2012.

[28] R. E. Wright, “Logistic regression.”, 1995.

[29] M. A. Hearst, S. T. Dumais, E. Osuna, J. Platt, and B. Scholkopf, "Support vector machines", *IEEE Intelligent Systems and their applications*, vol. 13, no. 4, pp. 18–28, 1998.

[30] L. Breiman, "Random forests", *Machine learning*, vol. 45, pp. 5–32, 2001.

[31] D. A. Reynolds *et al.*, "Gaussian mixture models.", *Encyclopedia of biometrics*, vol. 741, no. 659-663, 2009.

[32] U. Von Luxburg, "A tutorial on spectral clustering", *Statistics and computing*, vol. 17, pp. 395–416, 2007.

[33] D. Chicco and G. Jurman, "The advantages of the matthews correlation coefficient (mcc) over f1 score and accuracy in binary classification evaluation", *BMC genomics*, vol. 21, pp. 1–13, 2020.

List of Figures

List of Tables