2025

Analisi comparativa degli attacchi nei modelli di Centralized e Decentralized Federated Learning

Andrea Castellucci Elena Lucchi Sara Dublini

Abstract

Questo report fornisce un'analisi approfondita riguardo l'Apprendimento Federato Centralizzato (CFL) e l'Apprendimento Federato Decentralizzato (DFL), mettendo in luce le differenze cruciali tra i due approcci, con particolare attenzione al loro funzionamento e alle implicazioni per la sicurezza. Il DFL emerge come risposta alle limitazioni degli approcci centralizzati, offrendo miglioramenti in termini di comunicazione, resilienza e sicurezza.

Il report analizza in dettaglio le configurazioni e le caratteristiche dei sistemi FL e conduce un'analisi delle possibili fonti di vulnerabilità che riguardano ogni componente dell'architettura.

Maggiore attenzione è dedicata agli attacchi che minacciano entrambi i sistemi, ad esempio gli attacchi di avvelenamento o gli attacchi a gradiente. In particolare, gli attacchi di avvelenamento vengono classificati in base agli obiettivi (mirati e non mirati), ai metodi (avvelenamento dei dati e del modello), all'ambito (locali e globali) e alla tempistica con cui vengono attuati (durante l'addestramento o l'inferenza). Sono anche discusse le possibili strategie di difesa per contrastare tali attacchi.

Successivamente sono analizzati tre esempi pratici di attacco di avvelenamento per DFL, nello specifico l'attacco basato sulla similarità del coseno, che manipola la direzione degli aggiornamenti, l'attacco con massimo autovalore, che sfrutta la direzione di massima varianza dei dati per compromettere l'aggregazione, e l'attacco di avvelenamento del modello basato sull'angolo di Fisher, che utilizza la matrice di informazione di Fisher per guidare la manipolazione dei gradienti.

1. Introduzione

Negli ultimi anni, l'aumento della consapevolezza sulla protezione della privacy dei dati e l'applicazione rigorosa di regolamenti come il General Data Protection Regulation (GDPR) in Europa hanno reso i metodi centralizzati di elaborazione e archiviazione sempre più problematici.

Nel Machine Learning (ML) tradizionale, i dati devono essere raccolti e processati in un server centrale, secondo un approccio che solleva criticità in termini di privacy, sicurezza e costi di comunicazione.

Per superare queste limitazioni, è emerso il **Federated Learning (FL)**, un paradigma che consente l'addestramento collaborativo di modelli mantenendo i dati sui dispositivi locali e trasferendo solo gli aggiornamenti, come gradienti o parametri. Questo riduce significativamente i rischi legati alla condivisione di informazioni sensibili e migliora la scalabilità in scenari con grandi quantità di dispositivi connessi.

Un'ulteriore evoluzione è il Decentralized Federated Learning (DFL), che elimina la necessità di un server centrale per l'aggregazione del modello, distribuendo questo compito tra i partecipanti [1]. A differenza del Centralized FL, che dipende da un coordinatore centrale, garantisce maggiore robustezza e resilienza, evitando singoli punti di guasto e migliorando l'efficienza in ambienti con infrastrutture di rete limitate o intermittenti.

Questo report esplora i principi di funzionamento e analizza le configurazioni dei sistemi FL e DFL. Viene esaminata la vulnerabilità dei sistemi a vari tipi di attacchi e vengono discusse potenziali difese per mitigare tali minacce.

L'obiettivo è fornire una panoramica completa dei vantaggi, delle sfide e delle implicazioni di sicurezza di FL e DFL, identificando aree di ricerca promettenti per affrontare le nuove criticità introdotte da un approccio decentralizzato.

2. Corpo Principale

2.1 Centralized Federated Learning (CFL)

Federated Learning (FL) è un paradigma di apprendimento automatico distribuito che consente a molteplici entità (come dispositivi mobili, server aziendali o sistemi edge) di collaborare per addestrare un modello condiviso mantenendo i dati sensibili localmente. Questo approccio è stato inizialmente introdotto da Google per rispondere alla crescente esigenza di privacy e sicurezza, specialmente in applicazioni su larga scala che coinvolgono dati personali. FL supera i limiti dei modelli centralizzati tradizionali, in cui i dati devono essere trasferiti ad un unico server per l'addestramento.

In un sistema FL i dati non lasciano mai i dispositivi locali. Gli aggiornamenti del modello vengono trasmessi sotto forma di parametri o gradienti, garantendo che le informazioni sensibili rimangano protette. Questa caratteristica è particolarmente rilevante in settori come la sanità, la finanza e i servizi mobile, dove la privacy dei dati è una preoccupazione primaria.

In genere un server centrale coordina l'addestramento collaborativo di un modello globale. Il processo inizia con la distribuzione di un modello iniziale ad una federazione di client, ciascuno dei quali addestra sui propri dati locali e invia gli aggiornamenti dei parametri. Il server, a sua volta, aggrega gli aggiornamenti per aggiornare il modello globale e lo ridistribuisce. Questo ciclo viene ripetuto per più round fino a quando il modello globale non converge.

In questo approccio, chiamato anche **Centralized Federated Learning (CFL)**, il server centrale rappresenta il fulcro per l'aggregazione degli aggiornamenti e la creazione del modello globale.

La struttura introduce alcune vulnerabilità significative: in particolare, la presenza di un singolo punto di guasto rende l'intero sistema suscettibile a interruzioni, mentre i colli di bottiglia nella comunicazione possono limitare la scalabilità e l'efficienza dell'addestramento collaborativo.

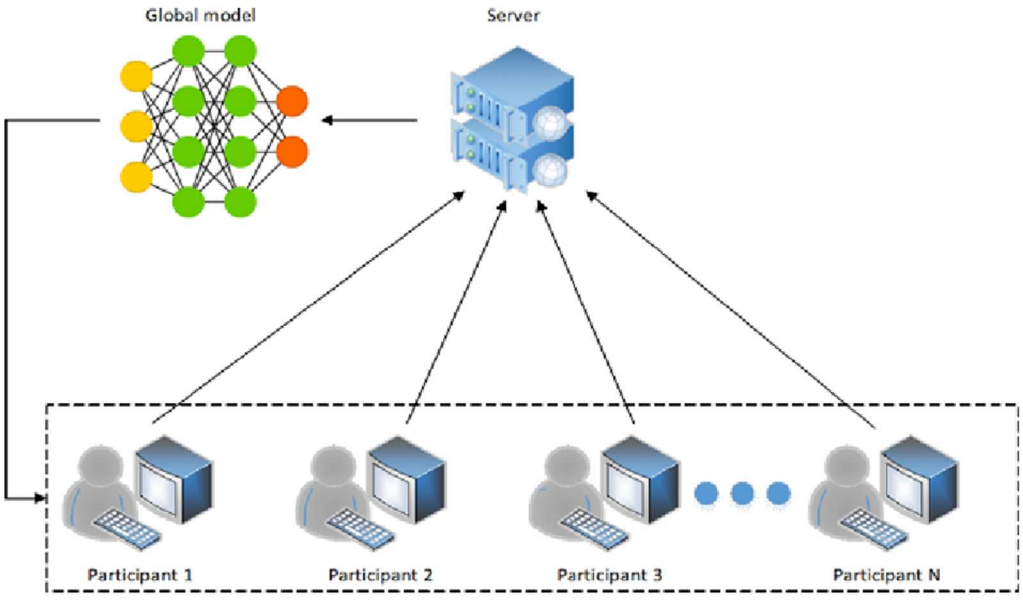

Figura 1: Framework generale di un'architettura FL

2.1.1 Funzionamento CFL

Il processo di funzionamento del paradigma CFL può essere esplicato in più fasi iterative:

1. Inizializzazione del modello globale

Un modello iniziale viene creato e inviato ai dispositivi partecipanti dal server centrale. Questo modello può essere completamente addestrato su un dataset preesistente o inizializzato casualmente.

2. Addestramento locale

Ogni dispositivo utilizza i propri dati locali per aggiornare il modello ricevuto. Questo avviene eseguendo un ciclo di addestramento autonomo che include la retropropagazione e l'ottimizzazione del modello.

3. Comunicazione degli aggiornamenti

Dopo l'addestramento locale, i dispositivi inviano al server centrale gli aggiornamenti del modello (ad esempio, i gradienti o i pesi aggiornati), che possono essere crittografati per una maggiore sicurezza.

4. Aggregazione degli aggiornamenti

Il server centrale combina gli aggiornamenti ricevuti utilizzando un algoritmo di aggregazione, come il Federated Averaging (FedAvg), e genera un modello globale aggiornato che rappresenta l'apprendimento combinato di tutti i dispositivi partecipanti.

5. Iterazioni ripetute

Il nuovo modello globale viene ridistribuito ai dispositivi, e il ciclo ricomincia fino a quando il modello raggiunge un livello desiderato di accuratezza o convergenza.

2.1.2 Categorie del Federated Learning (FL)

Federated Learning può essere classificato in tre principali categorie [2], in base alla distribuzione dei dati e alla relazione tra le entità partecipanti. Ognuna di queste categorie è progettata per risolvere problemi specifici legati ai dati e ai contesti applicativi.

a. Horizontal Federated Learning (HFL)

Horizontal Federated Learning si applica quando le entità partecipanti (ad esempio, dispositivi o organizzazioni) hanno dati che condividono lo stesso spazio delle caratteristiche, ma appartengono a utenti o entità diverse. In altre parole, i dataset hanno colonne (caratteristiche) simili ma righe (utenti) diverse.

In HFL, ogni dispositivo addestra localmente un modello utilizzando i propri dati e invia solo gli aggiornamenti dei parametri al server centrale per l'aggregazione. In questo modo si preserva la privacy, poiché i dati non vengono mai trasferiti, e migliorano sia l'accuratezza sia la capacità di generalizzazione del modello.

L'approccio risulta particolarmente efficace per applicazioni come la personalizzazione di app mobile (es. assistenti vocali), la raccolta di dati epidemiologici condivisi da ospedali o il miglioramento della qualità del servizio nei sistemi di raccomandazione online.

I principali limiti riguardano una distribuzione non uniforme dei dati (non-IID), che crea difficoltà nella convergenza del modello globale, e le risorse limitate che caratterizzano dispositivi mobili o IoT.

b. Vertical Federated Learning (VFL)

A differenza di HFL, le caratteristiche del dataset nel *Vertical Federated Learning* differiscono in modo significativo tra i client. Le entità partecipanti condividono utenti comuni, ma i dataset contengono caratteristiche diverse. In altre parole, i dataset hanno righe simili (stessi utenti) ma colonne differenti.

Pertanto, durante l'addestramento congiunto, è necessario prima effettuare un allineamento dei campioni dei dati di ciascun partecipante per ottenere i dati ripetuti degli utenti e poi procedere con l'addestramento sui dataset selezionati.

Questo approccio viene adottato in contesti come la collaborazione tra aziende in settori assicurativi e commerciali, o la ricerca congiunta tra istituti sanitari con dati complementari (analisi genomica e dati clinici).

I problemi più rilevanti risultano essere la sincronizzazione dei dati (gli utenti comuni devono essere identificati e sincronizzati tra i partecipanti senza violare la privacy) e la complessità computazionale data dalle tecniche di implementazione necessarie.

c. Federated Transfer Learning (FTL)

HFL e VFL richiedono che tutti i partecipanti abbiano lo stesso spazio delle caratteristiche o lo stesso spazio dei campioni per costruire un modello condiviso ed efficiente; tuttavia, in scenari più realistici, i dataset posseduti da ciascun partecipante possono essere altamente eterogenei. In questi contesti, il *Federated Transfer Learning (FTL)* emerge come una soluzione capace di costruire un modello globale efficace e accurato utilizzando una piccola quantità di dati, caratterizzati da campioni e caratteristiche non sovrapposti, e un numero limitato di etichette, rispettando le normative sulla privacy e sulla sicurezza dei dati.

FTL combina tecniche di apprendimento per trasferimento (Transfer Learning) e apprendimento federato, consentendo alle entità partecipanti di sfruttare conoscenze condivise anche in presenza di dati altamente eterogenei. Vengono utilizzate rappresentazioni astratte o funzioni pre-addestrate per trasferire conoscenze da un dominio all'altro, permettendo a ogni entità di affinare un modello condiviso sfruttando informazioni provenienti da altre entità, anche in assenza di una corrispondenza diretta dei dati.

I modelli FTL possono apprendere dal dominio di origine basandosi sulla somiglianza dei dati o dei modelli tra i partecipanti e, nella maggior parte delle applicazioni, le etichette nel dominio di origine vengono utilizzate per prevedere con precisione le etichette nel dominio di destinazione.

Questa modalità può risultare particolarmente utile per applicazioni come la collaborazione tra università e aziende, l'integrazione di dati ambientali da diverse fonti oppure la personalizzazione di esperienze di consumo attraverso settori diversi, ad esempio turismo e intrattenimento.

Tra gli aspetti negativi si possono citare sicuramente la difficoltà nel rendere compatibili i modelli per la collaborazione, a causa della disparità dei dati, e la definizione di un dominio condiviso tra dataset apparentemente indipendenti.

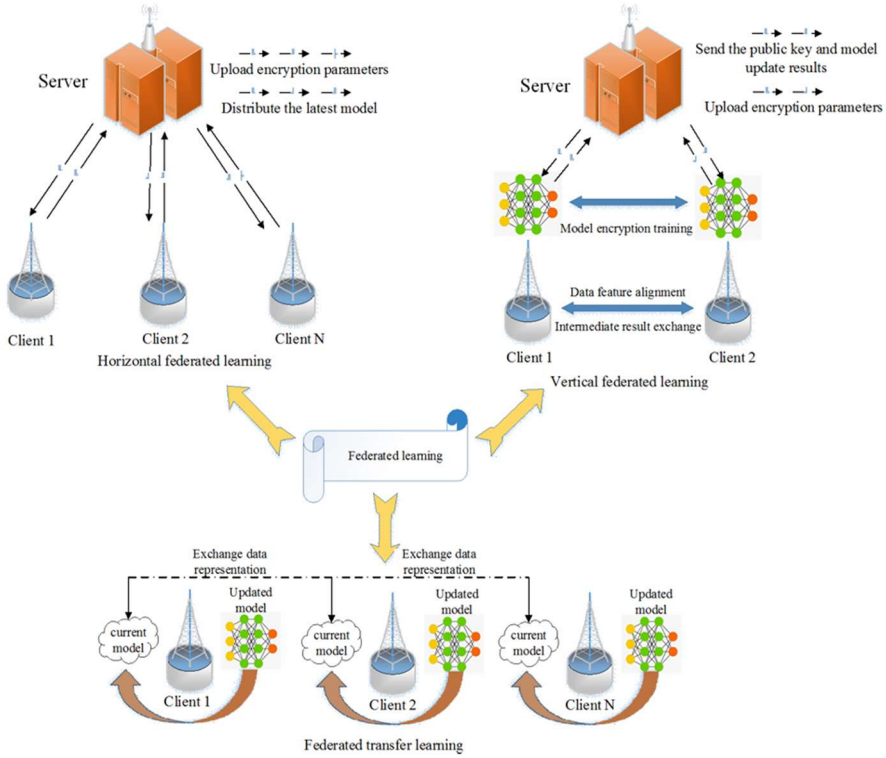

Figura 2: Categorie di FL

2.2 Decentralized Federated Learning (DFL)

Il DFL, noto anche come apprendimento federato distribuito, rappresenta un'evoluzione del Centralized Federated Learning (CFL) che elimina la dipendenza da un server centrale per l'aggregazione dei modelli.

L'aggregazione del modello è distribuita tra i nodi partecipanti, che si connettono e comunicano tra loro all'interno di una specifica topologia di rete. Ogni partecipante addestra il modello sui propri dati locali e scambia gli aggiornamenti con i nodi vicini. Successivamente, ciascun partecipante aggrega gli aggiornamenti ricevuti per aggiornare il proprio modello locale.

Il paradigma DFL consente un'interconnessione flessibile utilizzando varie topologie di rete: la scelta della topologia è cruciale, in quanto influenza sia i percorsi di comunicazione tra i nodi sia la velocità di convergenza del modello globale.

2.2.1 Caratteristiche dei sistemi DFL

Architettura di Federazione

L'architettura della federazione rappresenta l'organizzazione dei partecipanti e la distribuzione dei dati nel sistema. Si possono distinguere due principali tipi di federazione: cross-silo e cross-device [3].

La federazione *cross-silo* coinvolge organizzazioni o data center caratterizzati da un numero limitato di partecipanti, ciascuno con grandi volumi di dati a disposizione. Al contrario, la federazione *cross-device* comprende dispositivi come smartphone o dispositivi IoT, dove il numero di partecipanti è elevato ma ciascuno dispone di una quantità limitata di dati.

I partecipanti possono assumere diversi ruoli:

- il *trainer* addestra il modello locale sui propri dati;
- l'*aggregatore* raccoglie e aggrega gli aggiornamenti del modello dai nodi vicini;
- il *proxy* inoltra gli aggiornamenti tra nodi non direttamente connessi.

Nel contesto della decentralizzazione, è possibile distinguere diversi schemi in base alla distribuzione dei ruoli e delle responsabilità. Nel modello completamente decentralizzato (DFL), tutti i partecipanti condividono gli stessi privilegi e responsabilità, senza alcuna gerarchia. Nel semi-decentralizzato (SDFL), invece, un sottoinsieme di partecipanti assume il ruolo di aggregatore, con la possibilità di alternare la leadership tra i partecipanti. Infine, nel centralizzato (CFL), l'aggregazione del modello è gestita da un server centrale, come descritto nel paragrafo precedente.

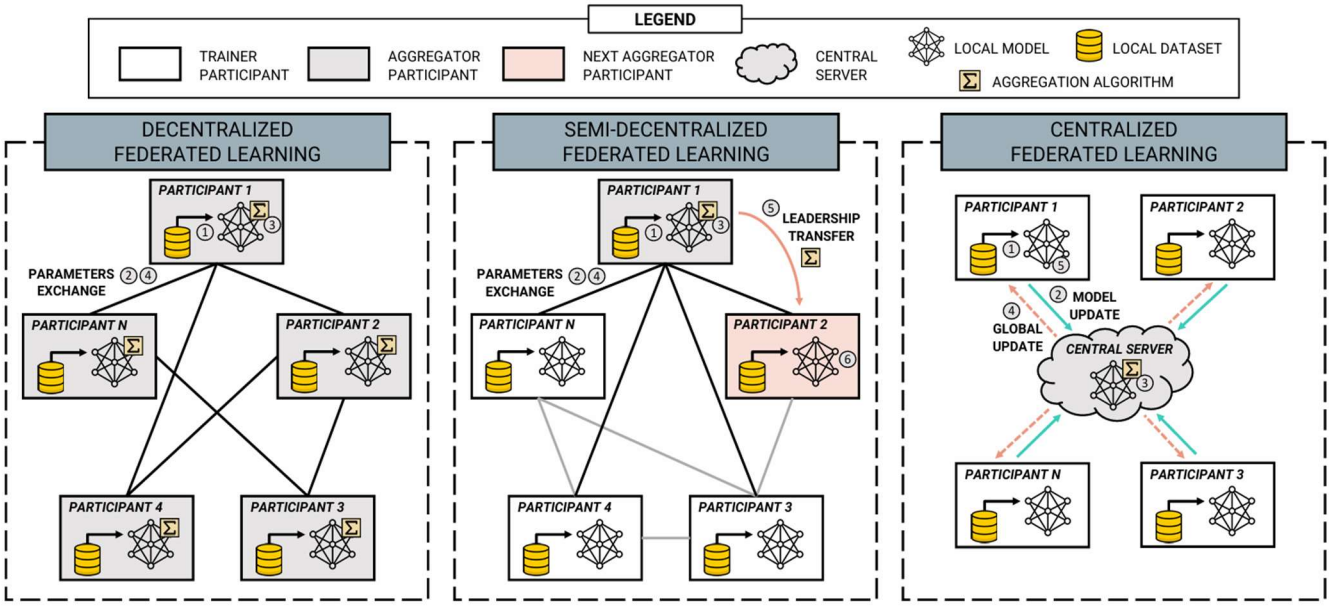

Figura 3: Confronto tra DFL, SDFL e CFL

La dinamica dell'addestramento e la convergenza del modello globale vengono influenzate anche dal modello di distribuzione dei dati, che può essere:

- *IID (Indipendente e Identicamente Distribuito)*, se i dati sono uniformemente distribuiti tra i partecipanti;
- *Non-IID*, se i dati presentano variazioni in termini di qualità, quantità e tipologia tra i partecipanti.

Topologia di Rete

La topologia della rete determina come i partecipanti sono connessi e comunicano tra loro. Le reti possono essere:

- *completamente connesse* se tutti i nodi sono connessi tra loro, offrendo una comunicazione efficiente ma un elevato overhead di comunicazione;
- *parzialmente connesse* se solo alcuni nodi sono connessi, riducendo l'overhead di comunicazione ma potenzialmente rallentando la convergenza del modello. Tra queste distinguiamo fra reti *star-structured* (un nodo centrale funge da hub per la comunicazione), *ring-structured* (i nodi sono disposti ad anello e comunicano con i nodi contigui), *random* (i collegamenti sono stabiliti in modo casuale);
- *clustering di nodi* se i nodi vengono organizzati in cluster, ciascuno dotato di un aggregatore dedicato. Questo approccio consente di bilanciare l'efficienza operativa con la riduzione dell'overhead di comunicazione.

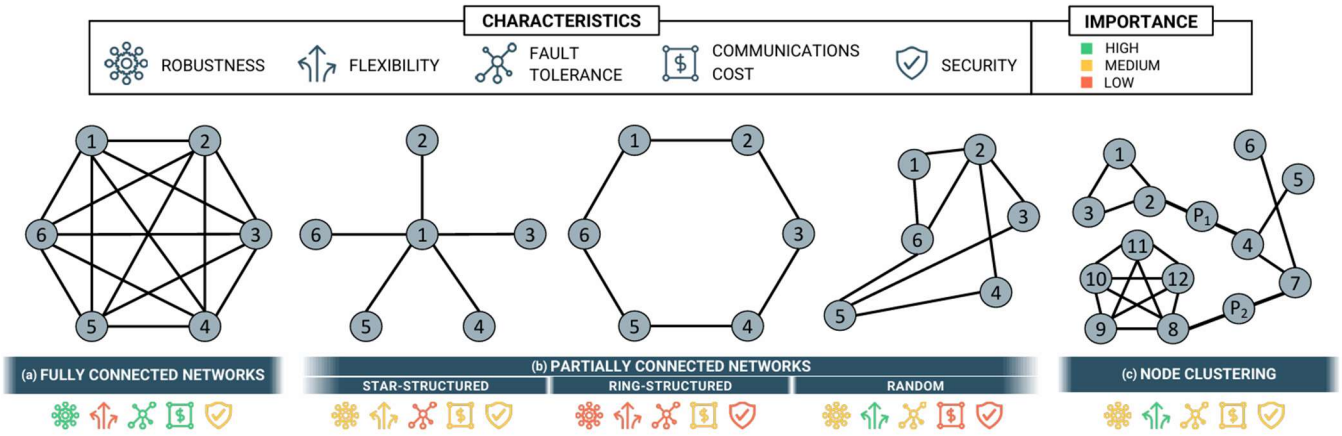

Figura 4: Topologie di rete DFL

La scelta della topologia influisce direttamente sulla robustezza del sistema, sui costi di comunicazione e sulla velocità di convergenza del modello globale.

Meccanismo di Comunicazione

Il meccanismo di comunicazione stabilisce come i partecipanti scambiano gli aggiornamenti del modello, influenzando direttamente l'efficienza e la coerenza del processo di addestramento.

Nella *comunicazione sincrona* tutti i nodi partecipano contemporaneamente alla comunicazione e all'aggregazione dei parametri del modello, garantendo un'elevata coerenza, ma al costo di un possibile rallentamento del processo. Al contrario, nella *comunicazione asincrona*, i nodi operano in modo indipendente, accelerando l'addestramento ma con il rischio di introdurre problemi di coerenza. Un approccio intermedio è rappresentato dalla *comunicazione semi-sincrona*, che cerca di bilanciare i vantaggi e gli svantaggi dei due schemi precedenti.

Per quanto riguarda l'architettura di rete, le comunicazioni *peer-to-peer (P2P)* permettono uno scambio diretto di informazioni tra i nodi, garantendo efficienza ma esponendo il sistema a vulnerabilità come intercettazioni o attacchi man-in-the-middle. Le comunicazioni *gossip*, invece, diffondono le informazioni in rete seguendo un modello simile al passaparola, risultando più robuste ai guasti, ma con una velocità inferiore rispetto al P2P.

Indicatori Chiave di Performance (KPI)

I KPI (Key Performance Indicators) sono fondamentali per valutare l'efficienza del sistema DFL e la loro misurazione aiuta a identificare le possibili aree di miglioramento. In particolare:

- i *KPI dei nodi di federazione* valutano le prestazioni dei nodi in termini di capacità delle risorse (con metriche come l'utilizzo di CPU/GPU, memoria e larghezza di banda di rete) e mobilità dei nodi (movimento dei nodi e impatto sulla stabilità della rete);
- i *KPI delle comunicazioni di federazione* misurano l'efficienza e l'affidabilità delle comunicazioni tra i nodi, in termini di flessibilità delle comunicazioni (capacità della rete di adattarsi a cambiamenti nella topologia o nella disponibilità dei nodi) e sovraccarico di rete (quantità di dati trasmessi e impatto sulle prestazioni);
- i *KPI dei modelli di federazione* valutano le prestazioni (in termini di accuratezza e precisione) e l'affidabilità (robustezza, sicurezza e resistenza ad attacchi o manipolazioni) del modello addestrato.

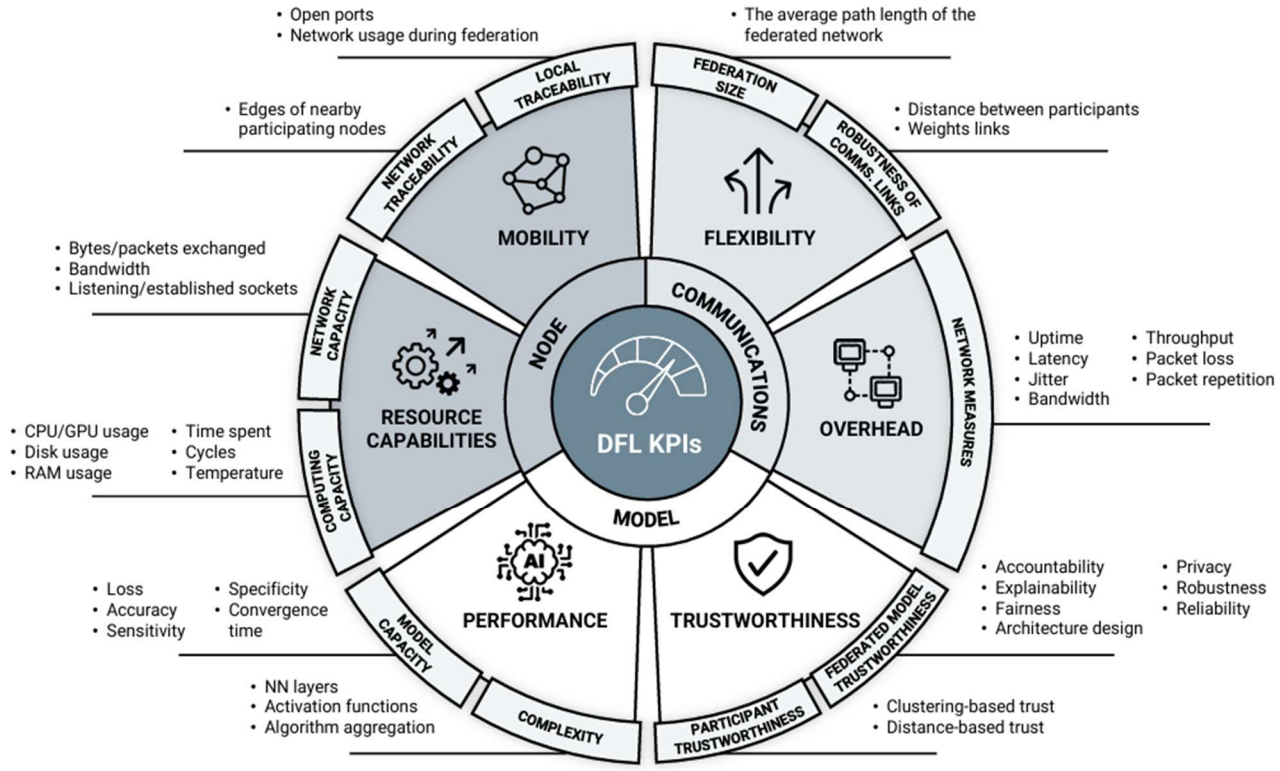

Figura 5: Indicatori chiave di performance per DFL

2.3 Confronto tra CFL e DFL in termini di sicurezza

CFL (Centralized Federated Learning) e DFL (Decentralized Federated Learning) non presentano distinzioni solamente riguardo le loro implementazioni tecniche, ma ottimizzano le problematiche reali negli specifici domini applicativi in modalità differenti, sfruttando i rispettivi punti di forza.

Nelle applicazioni pratiche, sebbene la struttura centralizzata di FL offra comodità nella gestione e nella sincronizzazione, può diventare un collo di bottiglia in termini di prestazioni ed esporre l'intera rete a rischi a causa di vulnerabilità, soprattutto in scenari che coinvolgono grandi volumi di dati e dispositivi partecipanti ampiamente distribuiti.

Al contrario, la natura decentralizzata del DFL offre vantaggi intrinseci nella protezione della privacy dei dati e nel miglioramento della sicurezza del sistema. L'operazione indipendente di ciascun nodo riduce i rischi e minimizza le superfici di attacco potenziali attraverso comunicazioni peer-to-peer dirette. Questo modello è particolarmente adatto per applicazioni che richiedono alti livelli di protezione della privacy dei dati, come la cooperazione tra multinazionali e la condivisione di informazioni mediche.

Distribuendo l'aggregazione dei parametri del modello tra nodi vicini anziché affidarsi a un'unica entità centrale, DFL offre diversi vantaggi rispetto a CFL.

- *Migliore tolleranza ai guasti*: la natura distribuita del DFL riduce il rischio di un singolo punto di errore, rendendo il sistema più resiliente.
- *Maggiore fiducia*: distribuendo la fiducia tra i nodi, il DFL riduce la dipendenza da un'unica entità centrale, mitigando i rischi associati alla fiducia in un singolo punto.
- *Riduzione dei colli di bottiglia*: la distribuzione del carico di lavoro e della comunicazione tra i nodi riduce il rischio di congestione e ritardi, migliorando le prestazioni generali del sistema.
- *Scalabilità automatica*: l'aggiunta di nuovi nodi al sistema è semplificata, dato che i nuovi nodi possono iniziare a comunicare e collaborare con i nodi esistenti senza la necessità di una riconfigurazione centralizzata.
- *Utilizzo efficiente delle risorse*: distribuendo la potenza di calcolo necessaria per l'aggregazione dei parametri del modello tra tutti i nodi partecipanti, il DFL utilizza in modo più efficiente le risorse disponibili.

Nel complesso, il Decentralized Federated Learning rappresenta non solo un progresso tecnologico, ma anche una nuova prospettiva sui modelli applicativi del machine learning che sicuramente giocherà un ruolo cruciale nei futuri scenari applicativi sensibili alla privacy e alla sicurezza.

Caratteristica	**FL Centralizzato**	**FL Decentralizzato**
Punto di Vulnerabilità	Server Centrale	Nessun Singolo Punto
Scalabilità	Limitata	Alta
Resilienza	Bassa	Alta
Complessità di Difesa	Inferiore	Superiore
Privacy	Dipende dal Server	Dipende dalla Rete

Nonostante questi vantaggi, introduce alcuni limiti.

A seconda della topologia di rete e dei meccanismi di comunicazione utilizzati, il DFL può comportare un aumento del sovraccarico di comunicazione rispetto al CFL. La mancanza di un server centrale complica poi l'ottimizzazione del processo di addestramento, rendendo più difficile garantire la convergenza e la generalizzazione del modello. Infine, garantire l'affidabilità e la sicurezza dell'IA in un ambiente decentralizzato pone sfide uniche in termini di protezione della privacy, rilevamento delle anomalie e prevenzione degli attacchi.

3. Attacchi e difese nei sistemi CFL e DFL

Vulnerabilità dei sistemi

CFL e DFL condividono vulnerabilità a diversi tipi di attacchi [4], che assumono forme specifiche nel contesto decentralizzato. Queste vulnerabilità possono essere sfruttate dagli avversari per compromettere il modello addestrato. A differenza dell'apprendimento automatico tradizionale, i sistemi devono affrontare tre possibili tipi di attaccanti:

1. i *client*, che essendo scelti casualmente tra migliaia o milioni possono essere dannosi senza che se ne possa rilevare la pericolosità solo tramite garanzie di sicurezza;
2. *l'aggregatore*, il quale può manomettere gli aggiornamenti dei client, alterare il processo di aggregazione e influenzare la visualizzazione del modello globale e dei parametri di compressione;
3. gli *esterni* o *intercettatori*, che cercano di compromettere il modello globale durante l'addestramento.

Esistono molteplici fonti di vulnerabilità, di seguito vengono analizzate alcune delle principali.

- *Comunicazione*: è possibile intercettare i modelli scambiati tra i diversi partecipanti, nonché il modello FL finale nella fase di distribuzione, e sostituirli con modelli dannosi.
- *Fuoriuscita di gradienti*: sebbene i dati non siano esplicitamente condivisi nel processo di addestramento, è comunque possibile per gli avversari rivelare informazioni delicate e persino ricostruire approssimativamente i dati grezzi.
- *Client compromessi*: i client possono osservare gli stati intermedi del modello globale e contribuire con gli aggiornamenti come parte della procedura di addestramento decentralizzata. Ciò crea opportunità per i client dannosi di manomettere liberamente il processo di addestramento.
- *Server compromesso*: il server può ispezionare tutti gli aggiornamenti dei gradienti in qualsiasi round di addestramento e manomettere il processo di aggregazione.
- *Algoritmo di aggregazione*: l'aggregatore dovrebbe incorporare meccanismi appropriati per rilevare gli aggiornamenti anomali dei client e scartarli.
- *Errore non dannoso*: fattori di sistema come una bassa larghezza di banda o una potenza di calcolo limitata possono causare errori in qualsiasi fase del processo FL. Tali errori possono escludere dal processo di addestramento i client con dati preziosi, con il risultato di un modello di bassa qualità.
- *Natura distribuita*: l'addestramento distribuito apre le porte ad attacchi collusivi e distribuiti, in cui più parti si accordano per lanciare un attacco coordinato.
- *Ambito dell'ambiente*: l'applicazione nella realtà di accordi tra le diverse parti partecipanti per definire l'ambito, lo scopo e le tecnologie utilizzate. Oltre ai vincoli specifici del dominio, la progettazione e l'applicazione dei protocolli di coordinamento possono essere difficili da definire e possono portare a situazioni di addestramento collaborativo poco robusto.

A differenza di CFL, il rilevamento di attività dannose in DFL pone sfide maggiori a causa dell'assenza di un'entità centrale che controlla l'intero processo.

3.1 Attacchi di avvelenamento

Negli attacchi di avvelenamento, gli avversari possono manomettere dati o modelli per interrompere l'efficacia e la robustezza di modelli benigni. Essi possono essere classificati in base a diversi criteri.

3.1.1 Criteri di classificazione

Obiettivi degli Attacchi

In base agli obiettivi, si distinguono attacchi non mirati e attacchi mirati.

Gli **attacchi non mirati** puntano a degradare in modo generico le prestazioni complessive di un modello, senza colpire specificamente alcun output. Questo tipo di attacco interrompe il processo di addestramento, rendendo il modello finale inaffidabile in diversi compiti.

Come discusso da Rathore e Basak [5], gli attacchi non mirati possono introdurre disturbi casuali come *rumore nei dati di addestramento*. Questi non richiedono una conoscenza della struttura del modello ma possono degradarne significativamente le prestazioni. Sfruttano tipicamente la sensibilità del modello ai dati di input, utilizzando modifiche lievi ma precise per degradare notevolmente le capacità di apprendimento e di generalizzazione del modello.

Un altro esempio di attacco non mirato è la generazione di *campioni avversari* visivamente o statisticamente simili ai campioni normali, ma progettati per indurre il modello a fare previsioni errate. Possono essere utilizzati non solo nella fase di test, ma anche nella fase di addestramento per interrompere il processo di apprendimento del modello. Lo scopo è creare campioni che siano sia abbastanza simili da passare inosservati sia abbastanza efficaci da attivare percorsi fuorvianti durante l'addestramento o il test del modello.

Gli **attacchi mirati**, al contrario, hanno come scopo quello di indurre il modello a prendere decisioni errate in condizioni o input specifici, con un obiettivo chiaro.

L'attacco di *backdoor* è un tipico esempio di attacco mirato, in cui l'aggressore inserisce specifici trigger nei dati di addestramento. Quando il modello incontra questi trigger durante la fase di inferenza, produce output errati predefiniti, mentre funziona normalmente in altre condizioni. Essendo attivati solo in condizioni specifiche, questi attacchi sono difficili da rilevare e contrastare durante la fase di addestramento, come descritto da Gu et al. [6].

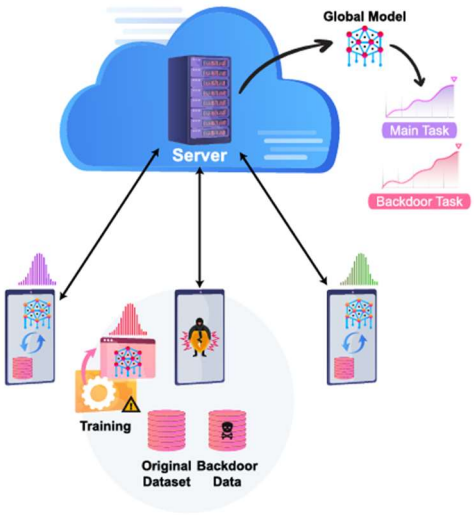

Figura 6: Attacco di backdoor

Un altro esempio è l'attacco di *classificazione mirata*, in cui l'obiettivo è far sì che il modello classifichi erroneamente le immagini di una persona specifica. Durante l'addestramento, l'attaccante può aggiungere perturbazioni sottili alle immagini della persona target, causando la costante errata classificazione di quella persona. Ad esempio, in un sistema di riconoscimento facciale, un attaccante può aggiungere rumore sottile alle immagini della persona target, impedendo al modello di procedere correttamente all'identificazione.

Nel contesto del Centralized Federated Learning (CFL), gli attacchi mirati si basano principalmente sulla manipolazione dei dati e vengono generalmente lanciati dal lato client. Al contrario, gli attacchi non mirati possono essere indipendenti dai dati, il che li rende eseguibili sia dal lato client che dal lato server.

Nel Decentralized Federated Learning (DFL), l'assenza di una chiara distinzione tra client e server permette a qualsiasi nodo di lanciare sia attacchi mirati che non mirati. Questa flessibilità amplia significativamente la superficie di attacco, aumentando il rischio complessivo.

Metodi di Attacco

In base alle diverse modalità di attacco, è possibile distinguere tra attacchi di avvelenamento dei dati (*data poisoning*) e attacchi di avvelenamento del modello (*model poisoning*).

Gli **attacchi di avvelenamento dei dati** compromettono il processo di apprendimento del modello iniettando campioni errati o malevoli nel set di dati di addestramento, portando a un degrado delle prestazioni o a output errati in presenza di specifici input.

Barreno et al. [7] hanno discusso per primi come gli avversari possano manipolare i modelli di apprendimento automatico attraverso l'introduzione di input malevoli. Ad esempio, gli aggressori possono aggiungere campioni di rumore specifici al dataset, disturbando l'apprendimento del modello e impedendogli di riconoscere correttamente i pattern nei dati, con un conseguente deterioramento delle prestazioni nella fase di test.

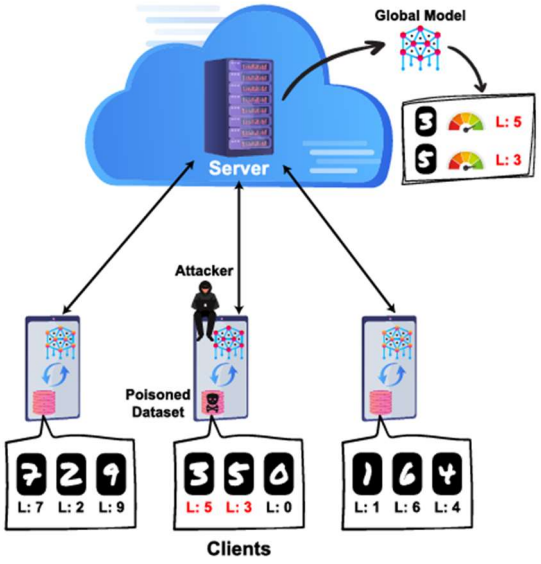

Figura 7: Attacco di avvelenamento dei dati

Una forma di attacco di avvelenamento dei dati è la *contaminazione dei campioni*, in cui gli attaccanti introducono campioni apparentemente normali ma in realtà malevoli nel dataset di addestramento, inducendo il modello ad apprendere caratteristiche dannose. Ciò può compromettere le prestazioni del modello durante la fase di test. Ad esempio, in un sistema di riconoscimento delle immagini, gli attaccanti possono aggiungere perturbazioni impercettibili alle immagini, inducendo il modello a produrre classificazioni errate. Un caso specifico è il riconoscimento dei segnali stradali: alterando leggermente le immagini dei cartelli nel set di addestramento, il modello può fallire nella classificazione corretta durante la guida autonoma, con potenziali gravi conseguenze sulla sicurezza stradale.

Dall'altro lato, gli **attacchi di avvelenamento del modello** consistono nella modifica diretta dei parametri del modello o degli aggiornamenti inviati al server, con l'obiettivo di degradarne le prestazioni.

Nel CFL, ogni client carica gli aggiornamenti del modello locale al server per l'aggregazione globale. Gli attaccanti possono intercettare e alterare questi aggiornamenti prima che vengano inviati, caricando *aggiornamenti manipolati* con perturbazioni lievi che, una volta aggregate nel modello globale, ne comprometteranno le prestazioni complessive.

Nel DFL, i partecipanti non solo possiedono conoscenza dei propri dati e modelli locali, ma possono anche accedere ai modelli dei nodi adiacenti. Questa maggiore visibilità consente ai nodi malevoli di ottimizzare strategicamente i loro attacchi, rendendoli più sofisticati e difficili da rilevare.

Un'altra forma di attacco di avvelenamento del modello è la *manipolazione dei parametri*, in cui gli attaccanti alterano direttamente i pesi del modello aggregato sul server. Questo tipo di attacco è particolarmente insidioso poiché non richiede il controllo di più partecipanti, ma solo un'operazione diretta sul server centrale.

Ambito degli Attacchi

In base all'ambito, si possono classificare attacchi locali e attacchi globali.

Gli **attacchi locali** sono comuni nei sistemi DFL e si caratterizzano per il loro ambito limitato, coinvolgendo solitamente pochi partecipanti o un singolo client. Questi interrompono il modello globale attraverso gli aggiornamenti dannosi caricati dai nodi compromessi. Come descritto da Zhu e colleghi [8], sebbene l'impatto diretto di questi attacchi sia limitato, un attaccante può comunque influenzare significativamente il modello globale caricando un grande numero di aggiornamenti dannosi in modo continuo o in momenti critici. Gli attacchi locali dimostrano inoltre la loro capacità distruttiva in compiti di classificazione delle immagini e riconoscimento vocale. Ad esempio, gli attaccanti potrebbero manipolare alcuni nodi per caricare dati di immagine con bias specifici, riducendo significativamente l'accuratezza del modello globale nel riconoscere determinate categorie. Nei sistemi distribuiti di riconoscimento vocale, i clienti controllati dagli attaccanti potrebbero caricare dati vocali rumorosi, causando al modello errori nel riconoscimento di determinati accenti o velocità di discorso.

Gli **attacchi globali** interessano un ambito più ampio, coinvolgendo un maggior numero di nodi, e provocano danni più gravi al modello. Questi attacchi rappresentano una minaccia maggiore per l'intero sistema FL distribuito, poiché possono influenzare l'operatività complessiva. Ad esempio, gli attaccanti possono manipolare più nodi per introdurre bias sistematici che, durante il processo di aggregazione del modello, porterebbero a un calo significativo delle prestazioni, rendendo il modello inaffidabile in vari compiti.

Tempistica dell'Attacco

Secondo il momento in cui avvengono, gli attacchi possono essere suddivisi in attacchi durante la fase di addestramento e attacchi durante la fase di inferenza.

Gli **attacchi durante la fase di addestramento** mirano a manipolare il processo di apprendimento attraverso operazioni di avvelenamento dei dati o del modello, influenzando i risultati di tale processo. Una caratteristica significativa è il loro impatto persistente: una volta che il modello è compromesso durante l'addestramento, il modello finale porterà sempre con sé questo pregiudizio, rendendo difficile la correzione nelle fasi successive. Ad esempio, attaccanti malintenzionati potrebbero spostare sistematicamente i confini decisionali dell'algoritmo manomettendo i dati di addestramento, inserendo notizie false, guidando così gradualmente il modello a sviluppare pregiudizi senza che ciò venga notato.

Gli **attacchi durante la fase di inferenza** manipolano le previsioni o le decisioni del modello, facendogli produrre specifici risultati errati. Una caratteristica notevole è la loro immediatezza: questo tipo di attacchi si attiva solo sotto condizioni specifiche, permettendo al modello di operare normalmente in altre circostanze. Tali attacchi sfruttano le incertezze del modello o le parti insufficientemente addestrate, utilizzando input progettati con cura per innescare decisioni errate. Ad esempio, introdurre perturbazioni con schemi specifici può causare il fallimento del modello in determinati scenari, mentre esso appare normale nei test di routine.

In tabella sono sintetizzati i criteri sopra descritti, includendo un esempio per ciascuno di essi.

Criteri di classificazione	Tipologia	Descrizione	Esempio
Obiettivi degli attacchi	**ATTACCHI MIRATI**	Inducono il modello a decisioni non corrette in condizioni specifiche	Introduzione di un rumore casuale nell'addestramento
	ATTACCHI NON MIRATI	Interrompono il processo di training, peggiorando le prestazioni complessive	Attacco backdoor: incorporazione di specifici trigger nei dati di training
Metodi di attacco	**AVVELENAMENTO DEI DATI**	Iniettano campioni errati o dannosi nei dati di addestramento	Aggiunta di email con label errate in sistema di classificazione di spam
	AVVELENAMENTO DEL MODELLO	Modificano direttamente parametri del modello o gli aggiornamenti sul server.	Caricare un modello addestrato localmente con sottili perturbazioni
Ambito degli attacchi	**ATTACCHI LOCALI**	Coinvolgono pochi o un singolo client, con impatto limitato ma significativo se persistente o tempestivo	Attacco ai nodi chiave di un sistema distribuito per caricare parametri errati
	ATTACCHI GLOBALI	Coinvolgono molteplici nodi, causando danni più gravi attraverso sforzi coordinati	Controllo di sensori in un sistema di guida autonomo per caricare dati incorretti
Tempistica dell'attacco	**DURANTE L' ADDESTRAMENTO**	Manipolano il training attraverso avvelenamento dei dati o del modello	Campioni fuorvianti nei dati di addestramento per disturbare l'apprendimento di modelli NLP
	DURANTE L'INFERENZA	Manipolano gli output del modello durante l'inferenza, causando specifici errori	Aggiunta di rumore specifico per ingannare i sistemi di riconoscimento veicoli in guida autonoma

Sebbene il DFL migliori la tolleranza ai guasti ed elimini il rischio di un singolo punto di vulnerabilità, espande significativamente la superficie di attacco, esponendo il sistema a una gamma più ampia di vettori di attacco. Di conseguenza, in termini di robustezza contro gli attacchi, il DFL non offre necessariamente un vantaggio rispetto al CFL.

3.1.2 Meccanismi di difesa contro attacchi di avvelenamento

La maggior parte dei metodi di difesa sono progettati per l'impostazione centralizzata, che è dotata di un processo di aggregazione del modello meno complesso. In DFL, la mancanza di un'entità di controllo centrale e l'aumento della gamma di vettori di attacco pongono delle sfide nello sviluppo di strategie di difesa.

Gli approcci per mitigare gli attacchi di avvelenamento possono essere di cinque categorie.

1. *Aggregazione Byzantine-robusta*: questi metodi mirano a proteggere l'aggregazione del modello dagli aggiornamenti dannosi, impiegando tecniche come la mediana per coordinate (COMED), filtraggio vettoriale e approcci basati su regolarizzazione o decomposizione. Tecniche come Krum e Multi-Krum sono esempi di aggregazione robusta, che selezionano aggiornamenti client basati sulla somiglianza con altri aggiornamenti, riducendo l'impatto degli attacchi. L'approccio basato sulla regolarizzazione mira a minimizzare l'influenza di aggiornamenti dannosi attraverso tecniche predittive come AFA.
2. *Rilevamento delle anomalie*: questi sistemi cercano di identificare aggiornamenti sospetti tramite metodi come Error Rate-based Rejection e Loss Function-based Rejection, che rimuovono gli aggiornamenti che peggiorano la prestazione del modello. Tuttavia, questi metodi possono violare la

privacy dei dati. Altri approcci, come FoolsGold e FLDetector, si concentrano sulla similarità del gradiente o sui contributi passati del client per rilevare anomalie.

3. *Difesa con target mobile (MTD)*: l'approccio MTD crea confusione per gli avversari malintenzionati introducendo una superficie di attacco dinamica e in continua evoluzione attraverso varie dimensioni del sistema. Il suo obiettivo è quello di aumentare l'incertezza e complicare gli attacchi. Un esempio è il protocollo Voyager, che varia la topologia di rete per migliorare la resilienza contro gli attacchi.
4. *Tecniche ibride*: queste combinano l'aggregazione e il rilevamento delle anomalie. Ad esempio, approcci come FLTrust e Trusted DFL usano modelli di riferimento per confrontare gli aggiornamenti e valutare la loro affidabilità. Altri metodi, come FedInv, utilizzano invece tecniche di inversione del modello per migliorare la sicurezza ma senza compromettere la privacy.
5. *Post-aggregazione*: intervengono dopo l'aggregazione del modello, modificando gli aggiornamenti per rimuovere o correggere eventuali danni. Esempi includono la potatura dei neuroni per attacchi backdoor e l'aggiunta di rumore gaussiano per migliorare la privacy.

In sintesi, sebbene ci sia un ampio lavoro in corso per migliorare la robustezza dei modelli in Federated Learning, gran parte delle ricerche si concentra sul Centralized Federated Learning (CFL), trascurando le peculiarità del DFL e le sue sfide legate alla decentralizzazione. Inoltre, molte di queste tecniche richiedono un alto carico computazionale e possono essere vulnerabili in ambienti eterogenei o con attacchi sofisticati.

3.2 Attacchi a gradiente

Gli attacchi a gradiente rappresentano una minaccia significativa per la sicurezza del Federated Learning (FL). Infatti, l'utente malintenzionato ottiene l'accesso ai gradienti del modello trasmessi dai client e li utilizza per ricostruire i dati di addestramento privati memorizzati sui client. Questo attacco sfrutta le tecniche di inversione del modello, consentendo all'attaccante di dedurre i dati originali dai parametri del modello e dai gradienti.

Deep Leakage from Gradients (DLG) è un noto attacco a gradiente che può compromettere sia il Centralized Federated Learning (CFL) sia il Decentralized Federated Learning (DFL).

In CFL, un server malintenzionato può raccogliere facilmente i gradienti del modello da tutti i client durante il processo di aggregazione e poi utilizzare tali informazioni per eseguire un attacco DLG e recuperare i dati privati dei client. In DFL, anche se non esiste un server centrale, un client malintenzionato può comunque tentare di recuperare i dati di addestramento privati dei suoi vicini; tuttavia, la sua natura decentralizzata rende questo compito più difficile rispetto a CFL.

3.2.1 Meccanismi di difesa contro gli attacchi a gradiente

Esistono diversi modi per difendersi dagli attacchi a gradiente: ad esempio si possono utilizzare tecniche di crittografia per proteggere i gradienti del modello durante la trasmissione oppure tecniche di privacy differenziale per aggiungere rumore ai gradienti del modello. Tali tecniche saranno riprese in seguito.

Guangxi Lu et al [9] hanno proposto un framework DFL progettato specificamente per resistere agli attacchi a gradiente, denominato *DEFEAT*. Utilizza una struttura di rete peer-to-peer (P2P) in cui i client comunicano solo con i loro vicini diretti, eliminando la necessità di un server centrale. In DEFEAT, ogni client addestra un modello locale diverso in modo indipendente: la mancanza di un modello globale unificato e la condivisione limitata dei gradienti rendono impossibile per un attaccante ottenere il gradiente esatto e ricostruire i dati privati.

DEFEAT opera in tre fasi principali:

1. Nella fase di *inizializzazione* un server configura la rete P2P e inizializza i parametri del modello per ogni client. Il server viene quindi disconnesso, assicurando un processo di addestramento decentralizzato.
2. Durante *l'addestramento locale* ogni client addestra il proprio modello locale sui propri dati privati senza condividerli con altri client.
3. Infine, nell'*aggregazione* dei parametri i client condividono i parametri del modello addestrato localmente con i loro vicini nella rete P2P e aggregano i parametri ricevuti per aggiornare i modelli locali.

L'efficacia di DEFEAT contro gli attacchi a gradiente è influenzata da diversi fattori, tra cui la topologia della rete P2P e il numero di round di comunicazione (D) in ogni round di addestramento.

Infatti, per quanto riguarda la topologia di rete si può notare che una rete più densa con più vicini per ogni client (R) aumenta la sicurezza. Man mano che R aumenta, i parametri locali vengono aggregati con un maggior numero di parametri del modello vicini, offuscando ulteriormente i dati e rendendo più difficile per un utente malintenzionato dedurre i gradienti individuali.

Inoltre, l'utilizzo di più round di comunicazione (D) in ogni round di addestramento può migliorare la resistenza di DEFEAT agli attacchi a gradiente. Con l'aumentare di D, i client condividono i parametri del modello in modo più estensivo, raggiungendo un consenso globale più rapido e rendendo più difficile per un utente malintenzionato isolare e sfruttare i gradienti specifici.

In sintesi, DEFEAT è un promettente framework DFL che offre una solida difesa contro gli attacchi a gradiente. Eliminando la necessità di un server centrale e limitando la condivisione dei gradienti, impedisce agli aggressori di ottenere i dati di addestramento privati dei client.

3.3 Altre tipologie di attacchi

Oltre alle tipologie sopra riportate, esistono altri attacchi che possono coinvolgere CFL e DFL.

Attacchi incentrati sugli algoritmi

Si tratta di una classe di attacchi in cui l'avversario viola l'integrità dell'algoritmo di aggregazione (supponendo che possa accedere al server o all'aggregatore in generale) o della pipeline di addestramento locale, sfruttando la *manipolazione delle regole di addestramento* o *compromettendo il calcolo distribuito*.

Nel primo caso, poiché l'autorità centrale dell'FL non ha alcun controllo sui client, gli avversari che controllano alcuni dei dispositivi partecipanti possono tentare di manipolare gli iperparametri di addestramento, come il numero di epoche locali, il tasso di apprendimento e la dimensione del batch, per impedire che il modello venga appreso del tutto.

In secondo luogo, l'attaccante può sfruttare a suo favore il problema di verificabilità, ossia l'incapacità di un client o del server di dimostrare agli altri partecipanti di aver effettivamente eseguito il comportamento desiderato: la progettazione del flusso di informazioni nel sistema può influenzare i risultati intermedi e renderli suscettibili agli attori malintenzionati.

Attacchi incentrati sulla federazione

La creazione di un sistema federato con proprietà di sicurezza ideali è un'impresa ardua a causa dei molteplici attacchi che prendono di mira la federazione stessa.

- *Attacchi di inferenza*: sfruttano l'analisi degli aggiornamenti del modello o del modello globale per dedurre informazioni sensibili dai dati di addestramento locali. Questi attacchi compromettono la privacy, anche se i dati grezzi non vengono mai condivisi. In DFL, la comunicazione diretta tra i nodi offre agli aggressori maggiori opportunità per raccogliere informazioni utili all'inferenza, amplificando il rischio di esposizione dei dati.
- *Attacchi di ricostruzione GAN:* gli attacchi delle reti generative avversarie (GAN) hanno caratteristiche simili agli attacchi di inferenza ma risultano più convincenti in quanto hanno la capacità di generare campioni artificiali che sono statisticamente rappresentativi dei dati di addestramento.
- *Attacchi free-riding:* consistono nel dissimulare intenzionalmente la partecipazione al processo FL per ottenere il modello finale senza contribuire effettivamente al processo di addestramento, ma inserendo aggiornamenti fittizi senza addestrare il modello con i loro dati locali.
- *Attacchi man-in-the-middle:* intercettano i modelli scambiati tra i client e il server e li sostituiscono con aggiornamenti dannosi del modello.

3.3.1 Altre possibili difese

Esistono diverse strategie di difesa per mitigare i rischi e proteggere il Centralized Federated Learning (CFL) e il Decentralized Federated Learning (DFL) dagli attacchi: la *difesa proattiva*, che rileva le minacce e i rischi correlati in modo prematuro, e la *difesa reattiva* che identifica l'attacco e adottando misure precauzionali.

Di seguito vengono illustrate diverse tecniche di difesa.

- *Ambiente di esecuzione attendibile*

Un Trusted Execution Environment (TEE) è definito come un ecosistema attendibile di alto livello per l'esecuzione di codice attestato e verificato. Il TEE stabilisce la fiducia digitale proteggendo i dispositivi connessi.

- *Pruning*

Riduce le dimensioni del modello di deep learning eliminando i neuroni per diminuire la complessità, migliorare l'accuratezza e disabilitare le backdoor.

- *Privacy differenziale*

La privacy differenziale funziona iniettando rumore statistico agli aggiornamenti del modello, in modo da garantire che nessun singolo record possa essere distinto in modo significativo dagli altri. Questo è utile nel caso in cui un utente malintenzionato cerchi di causare una modifica in alcuni campioni. Sebbene ciò introduca un compromesso tra privacy e accuratezza del modello, può offrire una protezione significativa per i dati locali senza compromettere il processo di addestramento globale.

- *Prove a conoscenza zero*

Le prove a conoscenza zero (ZKP) sono primitive crittografiche utilizzate per verificare le affermazioni da una parte (nota come prover) a un'altra parte (nota come verificatore) senza condividere o rivelare i dati sottostanti.

- *Adversarial Training*

Si riferisce a un problema di ottimizzazione minimax, in cui i campioni avversari e i parametri del modello vengono aggiornati alternativamente.

- *Federated multi-task learning*

Gestisce le sfide statistiche e di sistema come gli elevati costi di comunicazione, i ritardatari e la tolleranza agli errori. Nell'apprendimento multi-task, lo scopo è quello di apprendere modelli per più attività correlate contemporaneamente.

- *Riconoscimento dei client legittimi*

Questa strategia di difesa opera identificando gli utenti genuini e riducendo drasticamente il tasso di successo degli attacchi di avvelenamento anche quando sono coinvolti più avversari.

- *Federated distillation*

In caso di risorse di comunicazione limitate, lo scambio di parametri del modello diventa troppo costoso, in particolare per le moderne reti neurali profonde di grandi dimensioni. A questo proposito, la federated distillation è una soluzione FL convincente che trasferisce solo gli output del modello le cui dimensioni sono generalmente molto più piccole delle dimensioni del modello.

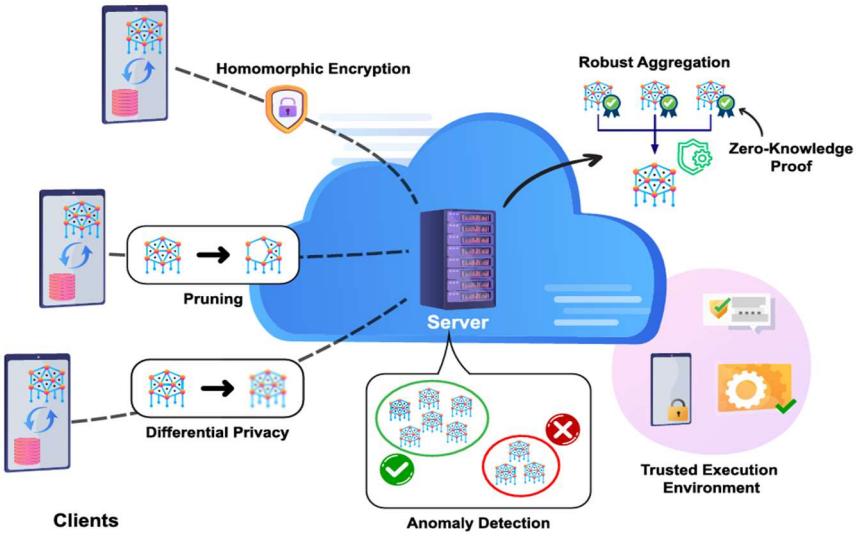

Figura 8: Meccanismi di difesa

4. Implementazioni pratiche di attacchi DFL

Attualmente la ricerca e l'applicazione pratica degli attacchi di avvelenamento nei sistemi DFL è significativamente carente, soprattutto rispetto ai sistemi CFL. Tuttavia, in questo report facciamo riferimento al progetto di ricerca di Runxi et al [10] che effettua una valutazione della sicurezza nei sistemi DFL attraverso la costruzione di modelli di attacco e la verifica dell'efficacia di tali attacchi in modelli simulati e reali.

Preparazione del dataset

L'esperimento viene condotto in un sistema di apprendimento federato composto da 50 nodi. Tra questi, alcuni nodi sono designati come nodi malevoli, responsabili dell'esecuzione degli attacchi. L'esperimento imposta un numero variabile di nodi malevoli (5, 10, 15) per osservare il loro impatto sulle prestazioni del sistema. Questi nodi affrontano 1200 round di addestramento e l'accuratezza più alta registrata durante questo periodo è definita come accuratezza massima globale.

Viene fatto uso del dataset CIFAR-10, composto da 60.000 immagini a colori con dimensioni di 32x32 pixel, suddivise in 10 categorie con 6.000 immagini per categoria. Questo dataset è ampiamente utilizzato nella ricerca sulla visione artificiale per compiti di riconoscimento delle immagini, in particolare perché fornisce un benchmark standard per la valutazione delle prestazioni dei modelli.

Il dataset CIFAR-10 è stato scelto per la sua diversità e complessità moderata, permettendo un test e una validazione efficaci di diversi algoritmi. Tale dataset viene elaborato attraverso una serie di processi, quali preprocessing dei dati, randomizzazione, segmentazione e allocazione dei dati.

Prima di avviare l'attacco, il dataset CIFAR-10 viene caricato e suddiviso in set di addestramento, validazione e test. Successivamente, i dati di addestramento vengono ulteriormente suddivisi in dataset per più utenti, ciascuno dei quali detiene un sottoinsieme indipendente di dati.

Durante ogni epoca di addestramento vengono applicate le strategie di attacco illustrate di seguito, le quali implementano il nucleo dell'algoritmo attraverso due funzioni principali:

1. *calculate_cos_angle(a, b)* è utilizzata per calcolare l'angolo coseno tra due tensori A e B. La funzione prima appiattisce i tensori, poi calcola il prodotto scalare tra i due e successivamente lo divide per il prodotto delle norme dei tensori per ottenere il valore del coseno dell'angolo. Questo valore del coseno riflette la somiglianza tra le direzioni dei due vettori; più è vicino a 1, più le direzioni sono simili.
2. *attack_cos_value(all_updates, model_re, n_attackers, dev_type)* implementa la strategia di attacco contro il modello con l'obiettivo di trovare un aggiornamento malevolo, in modo che l'angolo coseno tra l'aggiornamento e la direzione media di tutti gli aggiornamenti sia vicino a un determinato *target_angle*. I parametri della funzione sono:
    - *all_updates*: aggiornamenti di tutti i partecipanti;
    - *model_re*: parametri del modello di riferimento;
    - *n_attackers*: numero di attaccanti;
    - *dev_type*: specifica il tipo di disturbo ("unit vector", "sign", o "standard deviation").

4.1 Attacco Basato sulla Similarità del Coseno

L'attacco basato sulla similarità del coseno agisce in modo mirato, manipolando la direzione degli aggiornamenti per massimizzare il danno al modello globale. Invece di produrre aggiornamenti completamente errati, gli attaccanti mirano a fare in modo che i loro aggiornamenti siano "simili" a quelli onesti, ma con una direzione leggermente alterata.

L'attacco viene eseguito secondo una serie di passaggi chiave:

1. L'attaccante raccoglie gli aggiornamenti del modello da tutti i nodi e calcola la loro media. Questo determina la "*direzione media*" degli aggiornamenti onesti;

2. L'attaccante sceglie un angolo specifico (*target_angle*) rispetto alla direzione media, calcolato dinamicamente analizzando gli angoli coseno degli aggiornamenti degli altri nodi. In particolare, l'algoritmo ordina gli angoli coseno calcolati tra ciascun aggiornamento e la direzione media e, a seconda della differenza tra l'angolo massimo e minimo, seleziona un angolo specifico della lista ordinata come *target_angle*. Questo assicura che l'attacco sia ben calibrato;
3. L'attaccante genera un aggiornamento malizioso e poi lo modifica iterativamente, regolando un parametro *lambda* λ, in modo che l'angolo tra l'aggiornamento e la direzione media sia il più vicino possibile al *target_angle*;
4. Durante la manipolazione degli aggiornamenti, l'attaccante può usare diversi parametri come perturbazione per alterare la direzione dell'aggiornamento: il vettore unitario del modello di riferimento (*unit_vec*), il segno dei parametri del modello di riferimento (*sign*) o la deviazione standard di tutti gli aggiornamenti (*std*);
5. Infine, gli aggiornamenti manipolati vengono inviati insieme a quelli degli altri partecipanti.

L'attacco basato sulla similarità del coseno risulta particolarmente efficace per diversi motivi.

- **Furtività**: gli aggiornamenti maliziosi non sono molto diversi da quelli normali, rendendo difficile la loro individuazione attraverso metodi di difesa basati sulla rilevazione di anomalie evidenti.
- **Bypass delle difese**: l'attacco mira a manipolare la direzione degli aggiornamenti in modo tale da aggirare le difese che utilizzano metriche di distanza o perdita.
- **Mirato**: l'uso di un *target_angle* dinamico permette all'attacco di essere specifico e adattabile.
- **Adattabilità**: l'attacco è efficace contro diversi meccanismi di aggregazione, rendendolo versatile per vari scenari DFL.

Gli esperimenti hanno dimostrato che questo attacco è più efficace nel ridurre l'accuratezza del modello globale rispetto all'attacco di Shejwalkar [11], che ad oggi risulta essere il SOTA (State Of The Art) nell'ambito degli attacchi ai sistemi DFL. In particolare, con 10 nodi malevoli ottiene una precisione inferiore del sistema di 4,06 punti percentuali rispetto all'attacco Shejwalkar min-sum, corrispondente a un miglioramento del 10,6% nelle performance di attacco. Solo con 15 nodi maliziosi la performance appare relativamente più debole, con l'attacco cos che ottiene una precisione superiore di 6,92 punti percentuali rispetto all'attacco Shejwalkar min-sum, corrispondente a una riduzione del 21,2% nelle performance dell'attacco.

Si noti che, poiché l'obiettivo è testare le prestazioni dell'attacco, un valore inferiore di precisione indica una performance di attacco migliore.

4.2 Attacco con Massimo Autovalore

L'attacco di avvelenamento del modello basato sul massimo autovalore ha lo scopo di compromettere i sistemi DFL sfruttando la direzione di massima varianza dei dati**.** L'obiettivo principale è di influenzare il processo di aggregazione del modello globale manipolando i gradienti degli aggiornamenti, portando ad una riduzione della sua accuratezza.

Le principali fasi secondo cui viene implementato l'attacco sono:

1. L'attaccante, analogamente all'attacco con similarità del coseno, raccoglie gli aggiornamenti del modello da tutti i nodi e ne calcola un vettore che rappresenta la "*direzione media*";
2. A seconda del tipo di perturbazione definito dal parametro *dev_type*, che dipende dagli stessi parametri dell'attacco precedente, l'algoritmo calcola un vettore di deviazione che verrà utilizzato per modificare gli aggiornamenti;

3. L'attaccante calcola l'angolo coseno tra ogni aggiornamento e la direzione media, ottenendo una serie di valori che rappresentano la similarità di direzione tra ciascun aggiornamento e la media;
4. Gli angoli coseno vengono utilizzati per costruire una matrice (chiamata matrice C), sulla quale vengono calcolati autovalori e autovettori. L'autovettore associato al massimo autovalore indica la direzione di *massima varianza* nei dati;
5. L'attaccante seleziona l'autovettore associato all'*autovalore massimo* e utilizza una funzione sigmoide per elaborare l'angolo bersaglio. L'attaccante modifica l'aggiornamento del modello, aggiungendo una perturbazione nella direzione dell'autovettore di massimo autovalore;
6. L'attaccante utilizza un parametro *lambda* λ e lo modifica iterativamente fino a che l'angolo coseno tra l'aggiornamento modificato e la direzione media non supera un certo valore di soglia;
7. L'aggiornamento malizioso viene inviato al sistema DFL.

L'efficacia di questo attacco si basa su molteplici fattori.

- **Massima distorsione**: l'attacco manipola gli aggiornamenti nella direzione di massima varianza, dove le modifiche hanno l'impatto maggiore sul modello.
- **Bypass delle difese**: modificando i gradienti lungo le direzioni di massima varianza, l'attacco può aggirare le difese che si basano sulla rilevazione di semplici anomalie nei gradienti.
- **Stealth**: pur apportando modifiche significative al modello, l'attacco può rimanere difficile da rilevare poiché non introduce anomalie evidenti nei valori dei gradienti.
- **Impatto mirato**: concentrandosi sulla direzione di massima varianza, l'attacco minimizza le modifiche superflue e massimizza l'impatto sulle prestazioni del modello.

Per quanto riguarda le prestazioni, con 10 nodi attaccanti l'accuratezza dell'attacco con massimo autovalore è in media inferiore di circa il 25,0% rispetto a quella dell'attacco Shejwalkar min-max, il che si traduce in un miglioramento delle prestazioni del sistema di circa il 53,3%. Con 15 nodi attaccanti, l'accuratezza dell'attacco è in media inferiore di circa il 10,0% rispetto a quella dell'attacco Shejwalkar min max, rappresentando un miglioramento delle prestazioni di circa il 41,1%.

4.3 Attacco di Avvelenamento del Modello Basato sull'Angolo di Fisher

Questo attacco utilizza la matrice di informazione di Fisher (FIM), che misura la sensibilità dei parametri e quindi di fatto indica quanto un parametro contribuisce all'accuratezza del modello. I parametri più sensibili sono quelli che, se alterati, hanno un impatto maggiore sulle prestazioni. L'obiettivo principale dell'attacco è dunque quello di influenzare la direzione degli aggiornamenti individuando le parti più sensibili del modello, in modo da massimizzare il danno e ridurre l'accuratezza globale.

L'attacco segue il procedimento di seguito elencato.

1. L'attaccante raccoglie gli aggiornamenti del modello da tutti i partecipanti e calcola la "*direzione media*" degli aggiornamenti;
2. Poiché non si ha accesso alle seconde derivate necessarie per calcolare la matrice Hessiana, l'attaccante usa un'approssimazione della FIM. In particolare, l'algoritmo calcola l'approssimazione diagonale della FIM elevando al quadrato i gradienti;
3. L'attaccante calcola l'angolo coseno tra la FIM approssimata di ogni aggiornamento e la FIM approssimata media;
4. L'attaccante seleziona un angolo target (*target_angle*) basato sull'angolo coseno calcolato in precedenza, scegliendo l'aggiornamento con coseno minimo rispetto alla FIM approssimata media. Questo viene utilizzato come riferimento per la manipolazione degli aggiornamenti;

5. L'attaccante modifica iterativamente un aggiornamento malizioso, regolando il parametro *lambda* λ, con l'obiettivo di rendere l'angolo coseno tra l'aggiornamento e la direzione media più vicino possibile al *target_angle*. Durante questa fase, l'attaccante altera la direzione dell'aggiornamento secondo il tipo di perturbazione definito dal parametro *dev_type*, analogamente agli attacchi precedenti;
6. L'attaccante aggiunge una piccola quantità del prodotto tra la FIM approssimata media e la direzione media in direzione opposta a quella originaria, per modificare il modello di riferimento verso una regione dello spazio dei parametri che è dannosa per le prestazioni del modello globale;
7. L'aggiornamento manipolato viene inviato al sistema DFL.

L'efficacia di questo attacco si basa su diverse motivazioni.

- **Sensibilità dei parametri**: la FIM è una misura della sensibilità dei parametri del modello. Manipolando i gradienti in base alla FIM, l'attacco può massimizzare l'impatto dell'aggiornamento malizioso sul modello.
- **Massima distorsione**: l'attacco si concentra sulle zone del modello che sono più sensibili alle perturbazioni, massimizzando il danno causato dalle modifiche.
- **Aggiramento delle difese**: l'attacco manipola i gradienti utilizzando la FIM, rendendo difficile la rilevazione da parte delle difese basate sulla semplice analisi degli aggiornamenti.
- **Adattabilità**: l'uso della FIM e della deviazione standard degli aggiornamenti consente all'attacco di adattarsi a diversi contesti di apprendimento federato.

Gli esperimenti hanno dimostrato che l'attacco basato sull'angolo di Fisher mostra prestazioni superiori rispetto al metodo di riferimento Shejwalkar per diversi meccanismi di difesa. In generale, l'attacco con angolo di Fisher è risultato essere più efficace nel ridurre l'accuratezza del sistema, specialmente quando sono presenti 10 o 15 nodi malintenzionati. Quando ci sono 10 nodi malevoli, l'accuratezza dell'attacco Fisher è in media circa del 18,3% inferiore rispetto a quella dell'attacco Shejwalkar min-max, il che si traduce in un miglioramento delle prestazioni di circa il 37,5%. Con 15 nodi attaccanti, l'accuratezza dell'attacco Fisher è in media circa del 7,55% inferiore rispetto a quella dell'attacco Shejwalkar min-max, rappresentando un miglioramento delle prestazioni di circa il 31,2%. Rispetto all'attacco Shejwalkar min-sum, l'attacco Fisher si comporta leggermente peggio sotto alcuni meccanismi di difesa, ma rimane comunque forte, riducendo efficacemente l'accuratezza del sistema nella maggior parte dei casi.

Nel complesso, l'attacco Fisher si distingue per le sue eccellenti prestazioni attraverso molteplici meccanismi di difesa, in particolare sotto i meccanismi di difesa Krum e FedAvg. In questi due meccanismi di difesa, l'attacco Fisher può ridurre significativamente l'accuratezza del sistema con la maggior parte dei numeri di nodi attaccanti, dimostrando un chiaro vantaggio rispetto ai metodi di attacco Shejwalkar.

5. Conclusioni e Lavoro Futuro

Il Decentralized Federated Learning (DFL) si configura come un approccio promettente per l'addestramento collaborativo dei modelli di apprendimento automatico, presentando vantaggi significativi rispetto al Centralized Federated Learning (CFL), in particolare per quanto riguarda la resilienza, la scalabilità e la protezione della privacy.

Tuttavia, l'introduzione di DFL comporta anche nuove sfide di sicurezza che devono essere affrontate per consentire una sua implementazione affidabile. La sicurezza dei sistemi DFL dipende da diversi fattori, tra cui la topologia di rete, i meccanismi di comunicazione e gli algoritmi di aggregazione. Sebbene siano state proposte diverse tecniche di difesa, la loro efficacia dipende dall'attenta progettazione e implementazione del sistema.

I metodi di attacco riportati, come l'attacco basato sulla similarità del coseno, l'attacco con massimo autovalore e l'attacco di avvelenamento del modello basato sull'angolo di Fisher, mostrano forti capacità offensive, soprattutto in presenza di un maggior numero di nodi malevoli e di specifici meccanismi di difesa. Le loro prestazioni complessive risultano eccellenti, fornendo importanti riferimenti per ulteriori ottimizzazioni delle strategie di attacco e difesa nei sistemi FL. Inoltre, l'applicazione di successo di questi metodi offre prezioso supporto dati e una base teorica per future ricerche e applicazioni pratiche.

La ricerca futura su DFL dovrebbe infatti concentrarsi su diversi aspetti chiave per migliorarne la sicurezza, l'efficienza e l'adozione su larga scala. Un'area fondamentale riguarda lo sviluppo di algoritmi di aggregazione più robusti, capaci di resistere a una varietà di attacchi, anche in scenari con un numero elevato di partecipanti o con dati distribuiti in modo non indipendente (non-IID). Un altro obiettivo prioritario è l'ottimizzazione delle tecniche di privacy differenziale, affinché siano più efficienti, riducendo al minimo la perdita di accuratezza del modello pur garantendo una protezione della privacy adeguata.

Inoltre, è necessario progettare meccanismi di rilevamento delle anomalie avanzati, in grado di identificare tempestivamente comportamenti dannosi, anche in presenza di attacchi complessi e multi-fase. La ricerca deve anche concentrarsi sullo sviluppo di difese contro gli attacchi che potrebbero eludere i tradizionali meccanismi di rilevamento. Un altro aspetto cruciale è l'ottimizzazione della gestione dei nodi con risorse limitate, che è fondamentale per rendere i sistemi DFL scalabili ed efficienti, permettendo la partecipazione di dispositivi con capacità computazionali e di comunicazione limitate.

Sarà fondamentale creare framework modulari e scalabili che possano supportare diverse configurazioni di sistema e scenari applicativi, risolvendo le sfide legate all'efficienza e alla personalizzazione. La creazione di benchmark e dataset standardizzati per valutare l'efficacia delle soluzioni difensive proposte è altrettanto importante, poiché consente un confronto uniforme tra diverse metodologie.

Infine, la ricerca sugli attacchi avversari deve continuare per comprendere meglio il loro impatto su DFL e sviluppare contromisure adeguate, mentre l'esplorazione dell'apprendimento non supervisionato in DFL potrebbe aprire nuove opportunità per scenari in cui i dati non sono etichettati, ampliando ulteriormente le possibilità applicative di questo paradigma.

Considerazioni Finali

Affrontare le vulnerabilità di sicurezza nel DFL è essenziale per realizzarne appieno il potenziale come approccio sicuro e collaborativo nell'ambito dell'apprendimento automatico. Sebbene DFL offra vantaggi unici, la sua adozione diffusa richiede lo sviluppo di soluzioni innovative per risolvere le sfide di sicurezza, migliorare la resilienza e promuovere la fiducia in questo paradigma decentralizzato.

FONTI

[1] ”Sentinel: An Aggregation Function to Secure Decentralized Federated Learning”, Chao Feng, Alberto Huertas Celdrán, Janosch Baltensperger, Enrique Tomás Martínez Beltrán, Pedro Miguel Sánchez Sánchez, Gérôme Bovet, Burkhard Stiller

[2] “A survey on federated learning: challenges and applications”, Jie Wen, · Zhixia Zhang, Yang Lan2, Zhihua Cui2, Jianghui Cai, Wensheng Zhang.

[3] “Decentralized Federated Learning: Fundamentals, State of the Art, Frameworks, Trends, and Challenges”, Enrique Tomás Martínez Beltrán, Mario Quiles Pérez, Pedro Miguel Sánchez Sánchez, Sergio López Bernal, Gérôme Bovet, Manuel Gil Pérez, Gregorio Martínez Pérez, Alberto Huertas Celdrán

[4] N. Bouacida and P. Mohapatra, "Vulnerabilities in Federated Learning," in IEEE Access, vol. 9, pp. 63229-63249, 2021, doi: 10.1109/ACCESS.2021.3075203.

[5] P. Rathore and A. Basak,“Untargeted, targeted, and universal adversarial attacks and defenses on time series”, arXiv preprint arXiv:2101.05639,2020.

[6] T. Gu, B. Dolan-Gavitt, and S. Garg,“Targeted backdoor attacks on deep learning systems using data poisoning”, arXiv preprint arXiv:1712.05526,2017.

[7] M. Barreno, B. Nelson, R. Sears, A. D. Joseph, and J. D. Tygar, “Can machine learning be secure?”, Proceedings of the 2006 ACM Symposium on Information, Computer and Communications Security,2006.doi: 10.1145/1128817.1128824.

[8] C. Zhu, J. Ge, and Y. Xu,“Defend against poisoning attacks in federated learning”, in International Conference on Artificial Intelligence and Statistics,PMLR, 2021, pp. 2938–2948.

[9] Guangxi Lu, Zuobin Xiong, Ruinian Li, Nael Mohammad, Yingshu Li, Wei Li, “DEFEAT: A decentralized federated learning against gradient attacks, High-Confidence Computing, Volume 3, Issue 3, 2023.

[10] Cui, Runxi; Liu, Yunlong. *Novel Poisoning Attacks on Decentralized Federated Learning*. 2024, University of Zurich, Faculty of Economics.

[11] V. Shejwalkar and A. Houmansadr,“Manipulating the byzantine: Optimizing model poisoning attacks and defenses for federated learning”, in NDSS,2021.