Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE
Corso di Laurea in Informatica

Federated Learning Ibrido: Analisi e Performance

Relatore:
Prof.
Federico Montori

Presentata da:
Marco Perozzi

III Sessione
Anno Accademico 2022/2023

Abstract

Questo elaborato tratterà l'innovativa idea di Federated Learning Ibrido (FLI). Questo sistema mira a fondere insieme due concetti fondamentali ma opposti dell'intelligenza artificiale: il Federated Learning (FL) e il Machine Learning (ML) centralizzato. La discussione sul FLI sarà basata su una ricerca sperimentale che comporta l'implementazione di un ambiente che simuli l'ibridazione di queste due tecniche di apprendimento automatico.
In un mondo tecnologico in continua evoluzione in cui la decentralizzazione, la privacy dei dati e il libero arbitrio sono ormai un diritto degli utenti e un dovere per le imprese, non bisogna tuttavia venir meno ad un concetto basilare della tecnologia stessa: l'efficienza. L'obiettivo di questo studio sperimentale è quello di esaminare le performance di un sistema di apprendimento automatico ibrido basato sul FL, un paradigma avanzato di ML distribuito che consente a più dispositivi di addestrare in modo collaborativo un modello di IA, senza la necessità di condividere i dati grezzi[31].
L'elaborato presenterà nel dettaglio il FL nelle sue peculiarità, lati positivi e limitazioni. Si passerà a illustrare il dataset FEMNIST, progettato per un ambiente federato e basato sul riconoscimento di immagini. Successivamente verrà descritta l'idea del Federated Learning Ibrido e della sua elaborazione operativa e, infine, si esaminerà lo studio sperimentale e verranno analizzati i risultati prodotti.
L'ambizione della ricerca si è rivelata fondata, dato il vantaggio che porta unire un apprendimento centralizzato basato sull'efficienza con il FL basato sulla privacy. Le performance del sistema hanno mostrato un significativo aumento, tanto che il modello può registrare prestazioni con un miglioramento del 124%.

Introduzione

Il problema che pone il Federated Learning Ibrido (FLI) riguarda le sue performance rispetto al classico FL e questo elaborato mira esattamente a comprendere questo aspetto del nuovo sistema presentato. Per effettuare lo studio ci si è basati su analisi di simulazioni dell'ambiente. L'elaborato presenterà nel dettaglio tutti gli aspetti dell'ambiente di ricerca, sia le nozioni teoriche sia i tecnicismi progettuali, che hanno permesso di ottenere dei risultati consistenti.
Il primo argomento trattato sarà il contesto del progetto, basato sul FL appunto. Riguardo a questa tecnica verrà compreso perché è nato, come funziona, le classificazioni, quali sono i benefici, gli ostacoli e le sfide. Tra gli svantaggi verrà in particolare analizzato il problema del dataset non-IID.
Il secondo Capitolo ha come tema il dataset utilizzato per le simulazioni: FEMNIST. Verrà affrontato il concetto di database di immagini, saranno presentati i vantaggi di un dataset progettato per il FL e infine si vedrà la configurazione specifica di FEMNIST utilizzata per la ricerca.
Il Capitolo 3 spiegherà l'implementazione del concetto di FLI, dopo aver illustrato nel dettaglio l'idea alla base di questo sistema. Seguirà una spiegazione esauriente di come l'ambiente di ricerca in Python sia stato implementato.
L'ultimo capitolo ha come scopo l'analisi dei risultati. I dati prodotti saranno esaminati avvalendosi di grafici che mostrano il comportamento del sistema in base alle diverse variabili in gioco. La valutazione produrrà affermazioni e deduzioni riguardo il FLI. Infine si porrà l'interesse sui miglioramenti applicabili alla ricerca e al sistema reale.
L'elaborato terminerà con delle dichiarazioni conclusive e riassuntive sulle performance del sistema ibrido al centro dello studio.

Indice

Elenco delle figure

Capitolo 1

Stato dell'arte

Il contesto del progetto implementato si colloca all'interno di un campo emergente dell'apprendimento automatico, noto come Federated Learning (FL). Questa tecnologia rivoluzionaria sta cambiando il modo in cui pensiamo al Machine Learning (ML), offrendo un nuovo paradigma che pone l'accento sulla privacy e sulla decentralizzazione.
Sia i dati che la computazione sono decentralizzati nel Federated Learning, il che può portare a differenze significative e eterogeneità nei dati. Questa eterogeneità è una caratteristica fondamentale dei dataset non Indipendenti e Identicamente Distribuiti (IID), che sono spesso utilizzati nel Federated Learning. Questo è anche il caso del dataset utilizzato in questo progetto che implementa l'ibridazione del Federated Learning con il Machine Learning centralizzato.
In questo primo Capitolo verranno affrontati i due macro argomenti che sono alla base del progetto: il Federated Learning e la tipologia di dataset sul quale è avvenuto l'apprendimento federato, cioè i dataset non-IID. Si cercherà di analizzare le caratteristiche di ognuno, gli eventuali vantaggi o svantaggi e punti di partenza dai quali migliorare questi aspetti dell'apprendimento automatico, che presenta ampi margini di miglioramento.

1.1 Federated Learning

Il Federated Learning è una tecnica di apprendimento automatico che permette di addestrare modelli di intelligenza artificiale, in modo collaborativo e decentralizzato, su più

dataset locali senza condividere i dati tra le diverse parti coinvolte[30]. Proprio questa caratteristica di proteggere i dati dei partecipanti ha permesso a questa nuova tecnologia di svilupparsi e diffondersi sempre di più nel mondo dell'intelligenza artificiale. Infatti, sta diventando alla portata di tutti, sia per la possibilità di applicazione a contesti diversi, sia per i benefici che può apportare applicato ad un sistema preesistente.
Questo approccio, dunque, garantisce il rispetto della privacy, della sicurezza e dell'eterogeneità dei dati, proponendo diversi vantaggi rispetto al tradizionale ML ma, allo stesso tempo, pone delle sfide come la presenza di attori malevoli e la questione della disomogeneità dei dati utilizzati durante l'addestramento. Il FL può anche essere classificato in tre categorie in base alle caratteristiche che hanno in comune i diversi dataset dei partecipanti. Partiamo però dall'origine di questa tecnologia.

1.1.1 La risposta ai problemi di privacy

Per comprendere la nascita e il veloce sviluppo del Federated Learning è importante rendersi conto del contesto e degli aspetti del mondo tecnologico e digitale in cui esso è stato ideato e perché è così rivoluzionario. Con l'aumentare della condivisione da parte degli utenti di internet dei propri dati personali, considerando che quest'ultimi venivano acquisiti e memorizzati grezzi, da qualsiasi azienda che offriva un servizio online, sono incrementate anche le violazioni dei server delle suddette compagnie che ricevevano attacchi hacker, con la conseguenza che i dati dei clienti venivano esposti.
I data breach e la sempre più crescente consapevolezza di dover proteggere anche la propria identità digitale hanno spinto gli utenti di internet a richiedere maggiori diritti riguardo la privacy delle informazioni personali. Fino a questo momento non si era posto il problema e infatti le tecniche di Machine Learning erano centralizzate, ciò vuol dire che tutti i dataset locali appartenenti ad ogni cliente venivano raccolti in uno storage e l'apprendimento era eseguito su questa grande quantità di dati.
Proprio per questi motivi si è iniziato a pensare a una tecnica di apprendimento automatico decentralizzata, che risolvesse il sempre più in vista problema della privacy, così è stato ideato il Federated Learning. In questo caso i dati vengono tenuti al sicuro e memorizzati sul dispositivo locale dell'utente senza la necessità di doverli condividere con un server centrale. Il concetto di Federated Learning è stato presentato per la prima

volta nel 2016 sul blog di Google, e mostrava come Google Keyboard su Android salvasse la cronologia delle query nel dispositivo dell'utente per migliorarne i suggerimenti per l'iterazione successiva[26].
Dal momento in cui queste tecniche decentralizzate si sono dimostrate come la soluzione ai problemi di privacy per gli utenti e ai problemi di sicurezza e di memorizzazione per le aziende, l'apprendimento automatico federato si è sempre più diffuso, contraddistinto ed evoluto. Grazie al FL si è potuto ampliare l'applicazione dell'intelligenza artificiale a molti altri ambiti considerati non compatibili con il ML tradizionale, poiché si riteneva inadeguata la condivisione di determinati dati sensibili e strettamente personali, come ad esempio nel campo medico.

1.1.2 Operatività del FL

La differenza concettuale del FL con il ML è che in quest'ultimo i dati dai dispositivi vengono inviati al server centrale nel quale vengono memorizzati e avviene la computazione, mentre nel FL la computazione dal server centrale passa ai dispositivi degli utenti e, di conseguenza, i dati rimangono memorizzati localmente. Nell'apprendimento federato si ha quindi la decentralizzazione della potenza di calcolo, una distribuzione potenzialmente non uniforme dei dati e rispetto della privacy degli utenti partecipanti all'addestramento. In questo sistema, perciò, il server centrale si svuota delle mansioni principali dell'apprendimento automatico e, di fatto, si fa solo carico della comunicazione. Pertanto gli attori principali diventano i dispositivi che partecipano alla rete, chiamati client.
L'apprendimento nelle applicazioni di FL sfrutta un processo iterativo, dove ogni iterazione si chiama round, in cui i client sono i protagonisti dell'addestramento, con la partecipazione del server in alcune fasi. Lo svolgimento di un round si compone di queste fasi:

1. **Training centralizzato**
   Il server crea un modello globale e lo invia a tutti i dispositivi decentralizzati che parteciperanno al training locale

2. **Training locale**
   Ogni dispositivo riceve il modello dal server, che può essere preaddestrato o non addestrato. Questo viene allenato sui propri dati locali, contenuti nel set di addestramento, che non vengono condivisi né con gli altri client né con il server.
   La fase di apprendimento all'interno di uno stesso round può essere eseguito più volte. Il numero di ripetizioni sono chiamate epoche locali (local epochs), che equivale ad un iterazione completa dell'intero set di addestramento in cui ogni singolo dato viene elaborato dal modello[10].

3. **Aggregazione dei modelli**
   Successivamente i dispositivi inviano i propri modelli addestrati in locale al server. Esso quindi li aggrega per ottenere un modello globale condiviso. Ci sono diversi algoritmi di aggregazione che si possono implementare, il più utilizzato è il Federated Averaging (FedAvg), che effettua una media pesata dei modelli locali. Questo algoritmo è stato quello implementato nel progetto.

4. **Valutazione del modello**
   L'aggiornamento del modello viene poi mandato ai client che ne valutano le prestazioni e l'accuratezza rispetto alla precedente versione.
   Nel sistema ibrido in questa fase per tutti i round escluso l'ultimo viene utilizzato il set di validazione, che corrisponde ad un'esigua partizione del set di addestramento. Nell'ultimo round si effettua la valutazione finale impiegando il set di test.

5. **Ripetizione**
   Il processo appena descritto viene ripetuto finché il modello globale non raggiunge un livello di accuratezza soddisfacente oppure finché non vengono eseguite un predeterminato numero di iterazioni[7]. Per quanto riguarda l'ambiente del FLI tale procedimento viene ripetuto per 3 o 5 round.

Nel caso specifico di questo studio sul FL ibrido, sia l'addestramento del modello sia la sua valutazione in itinere e finale, sono eseguite dai client. Il server si occupa solo e soltanto dell'aggregazione dei parametri, nella fase di training, e delle metriche, valutate in fase di test.

1.1.3 Vantaggi della decentralizzazione

Compresa la differenza tra FL e ML tradizionale e analizzato il procedimento di un apprendimento federato si comprende comprendere come questo rappresenti un cambiamento significativo nel mondo dell'apprendimento automatico. La sua caratteristica principale cioè la decentralizzazione del processo e dei dati, offre una serie di vantaggi rispetto all'approccio tradizionale che hanno un notevole impatto su come vengono concepiti e implementati i modelli di intelligenza artificiale.

Maggiore scalabilità ed efficienza dei costi

Più dati potrebbero significare risultati migliori, tuttavia quando tutti i set di dati sono memorizzati in un unico server un addestramento di un modello su di essi potrebbe richiedere molto tempo. Poichè il FL permette di addestrare i modelli su tutti i dati contemporaneamente accelera l'implementazione dell'IA e migliora la scalabilità delle operazioni.
Oltretutto in quanto questa tecnica invia al server centrale solo i pesi addestrati vengono eliminati i costi di trasferimento dell'intero set di dati, riducendo anche il carico che gravava sulla rete attraverso la quale venivano condivisi i dataset locali.

Migliore adattabilità

I modelli di apprendimento federato possono essere applicati a nuove situazioni senza bisogno di riaddestrarsi. Inoltre, i miglioramenti e le conoscenze che questo metodo ottiene in un particolare ambito possono essere applicati ad un altro. Una tipologia di FL si basa esattamente su questo principio.

Superiore accuratezza e diversità dei dati

Quando i dati sono centralizzati e utilizzati per addestrare un modello, questi potrebbero non rappresentare accuratamente l'intero spettro al quale verrà applicato il modello. D'altro canto, addestrare un modello su dati decentralizzati provenienti da fonti diverse, beneficiando di un bacino più ampio poiché viene rispettata la privacy dei dispositivi

partecipanti, può migliorare la capacità del modello di generalizzare nuovi dati, gestire le variazioni e ridurre i bias[7].

Potenza di calcolo distribuita

Il FL permette di sfruttare la potenza di calcolo su più dispositivi, riducendo la necessità di enormi risorse di calcolo centralizzate. Risulta molto vantaggioso negli scenari in cui le risorse di calcolo sono limitate e, allo stesso tempo, estende l'applicabilità dell'apprendimento automatico a più attori. Infatti il ML centralizzato fino a questo momento era appannaggio delle poche società e istituzioni che si potevano permettere un'enorme capacità di calcolo centralizzata.

1.1.4 Sfide da affrontare

Il FL, in quanto una tecnologia sviluppata di recente, deve affrontare diverse sfide affinché possa affermarsi nel mondo dell'intelligenza artificiale e reggere il confronto con il più performante apprendimento centralizzato. Allo stesso tempo però, essendo una nuova tecnica di apprendimento automatico, presenta ampi margini di miglioramento.

Efficienza della comunicazione

Il FL comporta avere un numero considerevole di utenti, anche milioni in casi di estensione globale, in una sola rete. Il trasferimento dei messaggi quindi potrebbe diventare lento per colpa di diverse ragioni: ridotta larghezza di banda, mancanza di risorse o posizione geografica.
Per mantenere il canale di comunicazione efficiente, il numero di messaggi passati e la dimensione di quest'ultimi deve essere ridotta. È possibile ottenerlo utilizzando dei metodi di aggiornamento locale, per ridurre il numero di round; meccanismi di compressione del modello; training decentralizzato.

Privacy e protezione dei dati

È questa una delle questioni più importanti del FL. Sebbene i dati locali rimangono sul dispositivo dell'utente c'è il rischio che le informazioni vengano rivelate dagli aggior-

namenti del modello condivisi nella rete. Per risolvere il problema si possono usare delle tecniche per tutelare la privacy: aggiungere del rumore ai dati così da rendere difficile distinguere le informazioni effettive; crittografia omomorfica, eseguendo le computazioni su dati criptati; spartire le informazioni sensibili a più proprietari in modo tale da ridurre il rischio di violazioni della privacy.

Eterogeneità dei sistemi

Siccome un gran numero di client partecipano alle reti federate, tenere conto delle differenze di memoria, comunicazione e capacità computazionali è una sfida importante. Queste differenze posso essere gestite tramite tecniche di comunicazione asincrona, campionamento attivo dei dispositivi e tolleranza agli errori.

Eterogeneità statistica

Questo problema è posto dalle molteplici differenze dei dati presenti nelle memorie locali dei client. Ad esempio alcuni potrebbero avere immagini ad alta risoluzione, altri solo di bassa qualità. Questo denota che i dati potrebbero essere non IID in un ambiente di FL e ciò potrebbe causare problemi nelle fasi di strutturazione, modellazione e inferenza dei dati [33]. L'aspetto dei dati non-IID verrà esteso successivamente.

1.1.5 Classificazione del FL

Il FL può essere classificato in base a vari fattori, come il partizionamento dei dati, il tipo di modello, il meccanismo di privacy ed altre caratteristiche. Le tre tipologie principali di FL basati sulla suddivisione dei dati sono FL orizzontale, FL verticale e Federated Transfer Learning [31]. Questa classificazione viene effettuata in base a quale aspetto differenzia i vari dataset, o il campione o le sue caratteristiche, e di conseguenza quale approccio dell'apprendimento federato è più consono impiegare.

Horizontal Federated Learning

Il FL orizzontale o Horizontal Federated Learning (HFL) oppure detto anche sample-partitioned FL, può essere applicato in uno scenario in cui i set di dati locati in siti

differenti condividono uno spazio delle caratteristiche sovrapposto, ma differiscono nello spazio dei campioni, cioè istanze di dati differenti ma con le stesse caratteristiche. Un esempio sono i database delle banche, che conservano le stesse informazioni riguardo i propri clienti, con quest'ultimi che sono pressoché tutti diversi per ogni banca.
Per quello che riguarda la sicurezza di un sistema di apprendimento federato orizzontale si presuppone tipicamente che i partecipanti siano *onesti*, applicando una certa sicurezza contro un server, definito *onesto ma curioso*. Ciò vuol dire che solo il server può compromettere la privacy dei dati degli utenti. Per questo problema sono adottate tecniche crittografiche come l'homomorphic encryption, che forniscono un layer di sicurezza aggiuntivo sul server centrale per l'aggregazione dei parametri in fase di training [47].

Vertical Federated Learning

Il Vertical Federated Learning (VFL), invece, assume che i set di dati abbiamo lo stesso spazio dei campioni ma diverso spazio delle caratteristiche. Esso viene utilizzato nei contesti in cui gli ID degli utenti siano gli stessi ma i parametri dei client in ogni dataset siano diversi.
Un esempio è costituito dai database delle strutture sanitarie di una determinata area geografica, in cui uno stesso paziente può comparire in diversi archivi ma in ognuno di questi i dati raccolti sono diversi, poiché probabilmente ha effettuato visite o esami clinici differenti per ogni sede [31]. In questo caso, perciò, il FL verticale prevede di aggregare le differenti caratteristiche cliniche di ogni paziente preservandone la privacy, ottenendo così un quadro clinico completo per ognuno di loro, senza che le diverse strutture mediche condividano i dati sensibili riguardo la salute degli utenti coinvolti.
Un modello di sicurezza assume che i partecipanti siano *onesti ma curiosi*[29]. Questo riguarda la protezione dei dati e dei modelli delle parti coinvolte da possibili attacchi di inferenza, che mirano a ricostruire le informazioni private a partire dalle comunicazioni o dalle predizioni del modello. Per prevenire questi attacchi, si può adottare la crittografia omomorfica per la preservazione della privacy[37, 23], oppure, per agevolare la comunicazione sicura, si può definire un Semi-Honest Third Party che assumiamo non colluda con alcuna altra parte e fornisca prove di riservatezza[29].

Federated Transfer Learning

Il Federated Transfer Learning (FTL) è applicabile in un contesto in cui i dati delle parti differiscano sia nel sample space sia nel feature space. Questo metodo prevede il trasferimento di un modello pre-addestrato da un dominio di origine a un dominio di destinazione, dove i dati sono dispersi tra numerosi client. Il modello pre-addestrato viene poi messo a punto sul dominio di destinazione.
L'apprendimento federato per trasferimento può essere vantaggioso negli scenari in cui il dominio di origine contiene una notevole quantità di dati e competenze che possono essere applicate al dominio di destinazione [31]. Per presentare un esempio si può pensare di avere un grande set di immagini con valori di pixel ed etichette. Il modello pre-addestrato verrebbe istruito su questo set di dati per la classificazione delle immagini[31]. Tale modello verrebbe poi trasferito a un dominio di destinazione, dove i dati sono distribuiti tra più client. Il dominio di destinazione potrebbe essere un set di immagini mediche o di immagini di un'azienda privata. Il modello pre-addestrato verrebbe poi messo a punto sul dominio di destinazione, preservando la privacy e la sicurezza.
Per quel che riguarda la protezione dei dati e dei modelli da possibili attacchi di inferenza si può fare riferimento alle considerazioni fatte per il FL verticale.

1.2 Dataset non IID

Un aspetto cruciale del Federated Learning è la gestione dei dataset non IID, ovvero non Indipendenti e Identicamente distribuiti. Questa situazione si verifica quando i dati locali presentano delle differenze significative tra loro poiché ogni partecipante ha comportamenti e caratteristiche diverse. Il dataset non IID può influenzare negativamente le prestazioni e la convergenza dell'algoritmo globale [50], ma può anche essere sfruttato per apprendere dei modelli personalizzati per ogni utente. In questa sezione verranno analizzate le caratteristiche di un set di dati di questo tipo, quali sono le differenze con un dataset IID, quali sono i limiti e le conseguenze di un apprendimento federato con questo dataset e infine come superare gli ostacoli posti da esso.
Un dataset non IID è un insieme di dati in cui le variabili causali non sono indipendenti e/o non seguono la stessa distribuzione di probabilità. Questo può verificarsi perchè

all'apprendimento federato partecipano dispositivi o server diversi con set di dati altrettanto diversi. Tali differenze possono essere influenzate da fattori specifici come l'ubicazione geografica, la personalizzazione degli utenti, la variabilità temporale o la distribuzione delle classi.
Per una compresione completa dell'argomento è necessario prima osservare quando un dataset viene definito IID, in particolare cosa significa che esso sia Indipendente e Identicamente Distribuito, e di conseguenza quando queste caratteristiche vengono violate nel caso dei dataset non IID. Affiché un set di dati sia IID ogni punto dati al suo interno deve soddisfare queste due condizioni:

- **Indipendenza**
  Implica che l'occorrenza o il valore di un punto dati non fornisce alcuna informazione sull'occorrenza o sul valore di un altro punto dati. Presuppone, quindi, che i punti dati non siano influenzati l'uno dall'altro e che non vi sia alcuna struttura nascosta o correlazione tra di essi[12].

- **Identica Distribuzione**
  Presume che i punti dati condividano la stessa distribuzione di probabilità e che quindi questa sia omogenea per tutto il dataset. Ciò implica che le proprietà statistiche, come la media, la varianza e altre caratteristiche distributive, rimangano coerenti nell'intero set di dati. Pertanto, un dataset non dovrebbe contenere tendenze, poiché indicano che una singola distribuzione di probabilità non descrive tutti i dati[12].

Per semplificare la definizione si può affermare che un dataset IID è un insieme di dati in cui ogni punto dati è paragonabile al risultato di un lancio di un dado equilibrato: ogni faccia del dado ha la stessa probabilità di apparire e ogni lancio non dipende dai lanci precedenti[9].
Di conseguenza, un set di dati è definito non IID quando i dati non sono indipendenti o non sono distribuiti identicamente oppure quando entrambe le condizioni non si verificano nei dataset locali dei partecipanti al FL. In questi casi, quindi, è possibile che i dati non siano tutti statisticamente indipendenti e potrebbero esserci delle relazioni tra questi. La violazione dell'assunto di Indipendenza può portare a previsioni del modello distorte

o inaffidabili[12]. Se invece è l'ipotesi dell'Identica Distribuzione a non verificarsi allora esiste la possibilità che la distribuzione di probabilità non sia uniforme e che quindi alcuni dati si presentino con un occorrenza maggiore o minore rispetto ad altri nei diversi dataset, infrangendo l'omogeneità. L'inadempienza di questo assunto può introdurre un bias di campionamento, causando una scarsa generalizzazione dei modelli nei confronti di dati nuovi e mai osservati prima[12].
Vediamo ora nello specifico le conseguenze dell'utilizzo di un dataset non IID su un apprendimento tramite FL e se esistono delle soluzioni a questi problemi.

1.2.1 Conseguenze e soluzioni

Come accennato precedentemente, l'apprendimento federato su dataset così strutturati nella maggior parte dei casi può causare un deterioramento delle prestazioni, con un conseguente decremento dell'accuratezza del modello risultante al termine del training. Questo peggioramento dei risultati presuppone il paragone con un addestramento su un dataset IID, mantenendo uguali le configurazioni e gli algoritmi.
Questi risultati sono dimostrati da uno studio sperimentale che prova una riduzione dell'accuratezza fino a circa il 55% per reti neurali convoluzionali addestrate su dati non IID altamente distorti, avvalendosi del Federated Averaging (FedAvg) come algoritmo di aggregazione[50]. Questa diminuzione dell'accuratezza dell'addestramento può essere spiegata dalla divergenza dei pesi, che aumenta in modo direttamente proporzionale all'asimmetria della distribuzione dei dati, cioè man mano che il dataset diventa più non IID [50]. Per contestualizzare, la divergenza dei pesi misura la differenza dei pesi del modello risultante tra due processi di addestramento diversi, data la stessa inizializzazione dei pesi.
Questo impatto negativo che che i dati non IID possono avere sugli algoritmi di apprendimento automatico risulta nella produzione di risultati distorti o non affidabili, l'overfitting o l'underfitting dei dati[19]. I modelli di Deep Learning, in particolare, sono molto sensibili alla proprietà IID dei dati. Tra questi complessi modelli utilizzati nel Deep Learning ci sono anche le Convolutional Neural Network (CNN), che è anche il caso del modello implementato nel sistema di ibridazione del FL, che dipendono fortemente dai dati sui quali vengono addestrati.

Per affrontare questi problemi è possibile verificare la proprietà IID del dataset che si ha a disposizione semplicemente visualizzando i dati oppure tramite strumenti statistici, quali misurare la media e la deviazione standard. Se i dati risultano non IID ci sono diverse tecniche per migliorare tale caratteristica modificando il dataset, oppure c'è la possibilità di agire nel senso opposto, quindi sfruttare metodi che permettano di accrescere i risultati dell'apprendimento con l'originale dataset non IID.
Le soluzioni proposte sono le seguenti:

- **Campionamento dei dati**
  Questo metodo prevede la selezione di un sottoinsieme rappresentativo dei dati per l'addestramento dei modelli di FL[19], modificando la distribuzione dei dati nel dataset per renderla più equilibrata. Questa tecnica si può applicare a quegli insiemi di dati in cui l'asimmetria si riscontra a livello delle classi e quindi applicare l'oversampling, in cui si aumenta il numero di esempi della classe minoritaria, oppure l'undersampling, in cui si riduce il numero di esempi nella classe maggioritaria[25]. Scegliendo un sottoinsieme rappresentativo, si può garantire che i dati siano IID.

- **Incrementare i dati**
  L'aumento dei dati consiste nel generare nuovi esempi di addestramento a partire dai dati esistenti applicando trasformazioni casuali, come rotazioni o traslazioni[19]. Un'altra tecnica implica la generazione di dati sintetici, cioè prodotti a partire dai dati reali, usando tecniche che apprendono ciò che contraddistingue i dati già presenti nel dataset[32].
  Questi approcci possono contribuire a migliorare la proprietà IID del dataset aumentandone la diversità o la quantità dei dati di addestramento.

- **Normalizzazione dei dati**
  Questa tecnica prevede di pre-processare i dati per ottenere una distribuzione più uniforme così da avere una media il più possibile vicino a 0, rendendo quindi il dataset più IID[19].

- **Condivisione globale di dati**
  Strategia che è stata proposta per migliorare l'addestramento direttamente su dataset non IID creando un sottoinsieme di dati che è condiviso globalmente tra tutti

i dispositivi. Esperimenti mostrano che l'accuratezza può essere aumentata fino a circa il 30% con solo il 5% di dati condivisi globalmente, utilizzando in questo studio il dataset CIFAR-10.
In sostanza, si possono incrementare le prestazioni di un addestramento di FL con un dataset non IID senza modificarlo e, soprattutto, senza compromettere la privacy dei partecipanti, poiché i dati condivisi globalmente costituiscono un dataset separato rispetto a quelli locali dei singoli dispositivi[50].

Capitolo 2

Il dataset: FEMNIST

Questo capitolo verterà intorno al concetto di dataset e, in particolare, su quello con il quale sono stati addestrati i modelli di questa ricerca sul Federated Learning Ibrido: FEMNIST. Esso altro non è che un adattamento per apprendimenti di tipo federato del database EMINST (Extended MNIST), che a sua volta estende l'originale MNIST e infatti, FEMNIST sta per Federated Extended MNIST. MNIST sostanzialmente è un dataset di cifre scritte a mano utilizzato per l'elaborazione di immagini[42].
È determinante in primo luogo considerare l'importanza fondamentale che ricoprono i dati nel campo del ML. Infatti, l'apprendimento automatico dipende fortemente dai dati poiché proprio su questi si basa l'addestramento di un modello, che viene istruito a identificare determinati pattern nei dati. Nei progetti di FL, e in generale nel ML, si utilizzano tre diversi set di dati:

- Set di addestramento (training set) che è utilizzato per addestrare il modello
- Set di convalida (validation set), necessario per ottimizzare il modello
- Set di test (test set) che serve per valutare le prestazioni del modello

Oltretutto è importante che questi set di dati siano etichettati e classificati. L'etichettatura comporta l'assegnazione di etichette ai dati in base a ciò che rappresentano, mentre la classificazione determina le categorie all'interno delle quali i dati sono raggruppati[18]. Prendiamo l'esempio di MNIST: le etichette sono interi che vanno da 0 a 9 e definiscono quale numero rappresenti una determinata immagine; le classi sono sempre le dieci cifre,

che raccolgono tutte le immagini di ogni cifra in un solo insieme.
Successivamente all'introduzione generale e alla discussione sulla struttura fondamentale di un set di dati utilizzato per l'apprendimento automatico, si procederà con l'analisi dei dataset MNIST ed EMNIST, dai quali deriva FEMNIST.

2.1 Database di immagini: MNIST, EMNIST e simili

Il database MNIST (Modified National Institute of Standards and Technology[22]) è una vasta base di dati di cifre scritte a mano, comunemente usato per addestrare sistemi di elaborazione e riconoscimento di immagini, in particolare per addestrare e testare modelli nel campo dell'intelligenza artificiale. Esso è composto da 60,000 immagini per l'addestramento e 10,000 per la fase di test, dove ogni immagine ha una dimensione fissa di 28x28 pixel.
Questo dataset è stato ottenuto dalla combinazione dello Special Database 1 e Special Database 3 del National Institute of Standards and Technology (NIST) che consistevano in cifre scritte a mano, il primo da studenti della scuola superiore e l'altro da dipendenti stessi del NIST. Tali set di dati originari erano composti da immagini binarie 128x128, che sono state elaborate normalizzando la loro dimensione e centrandole, fino ad ottenere immagini in scala di grigi di 28x28 pixel.
L'evoluzione del set di dati MNIST è costituito da Extended MINST (EMNIST), che ne è diventato il successore. Esso infatti è un database ancora più vasto che comprende immagini in scala di grigi 28x28 pixel, oltre che delle cifre, anche di tutte le lettere dell'alfabeto maiuscole e minuscole scritte a mano[42]. La totalità dei dati include ben 814,255 caratteri, divisi in 62 classi, con la possibilità anche di usufruire di sottoinsiemi dell'intero dataset EMNIST[6]. Tutti questi dati derivano dallo Special Database 19 del NIST. Come per MNIST le immagini del database originario sono state elaborate allo stesso modo e fornite nello stesso standard, di conseguenza, gli strumenti che funzionano con il dataset MNIST, più datato e di minor dimensione, funzioneranno probabilmente senza modifiche anche con EMNIST[42].
Un altro set di dati che si inserisce nella cornice dell'addestramento di modelli sul rico-

noscimento di immagini è Fashion-MNIST, composto da immagini 28x28 in scala di grigi di articoli d'abbigliamento di Zalando. Composto, come MNIST, da un training-set di 60,000 esempi e un test-set di 10,000, associati ad un'etichetta tra 10 classi diverse. Il team di ricercatori affermano di averlo ideato perché MNIST è troppo semplice, con reti neurali convoluzionali è facile ottenere un accuratezza molto elevata, ed è abusato[46]. L'obiettivo è quello di proporre un'alternativa più sofisticata, al fine di elevare il parametro di confronto sulle prestazioni di un modello.
Altri esempi sono: Kuzushiji-MNIST, che raccoglie immagini di caratteri dell'alfabeto giapponese scritti a mano, ed ha la stessa struttura e formato di MNIST[5]; CIFAR-10, dataset impiegato per la classificazione di immagini e il riconoscimento di oggetti, contenente 10 classi di immagini tra oggetti e animali[38].

2.2 FEMNIST

Il dataset che è stato utilizzato in questo studio si chiama Federated Extended MNIST (FEMNIST) che è un adattamento del dataset EMNIST per ambienti federati, in cui è compreso il Federated Learning. Questo adeguamento di un dataset ad una versione federata altro non è che una suddivisione del set di dati, sia del set di addestramento che del set di test, in un numero di partizioni che rappresentano il numero massimo di dispositivi che possono partecipare ad un apprendimento federato.
In particolare, FEMNIST è stato rilasciato da LEAF, un framework di benchmarking per l'apprendimento in ambienti federati, con applicazioni che includono anche il FL. Il set di dati fornito è partizionato in 3,550 dispositivi con un totale dei campioni pari a 805,263, quindi la media dei campioni per dispositivo è pari a 226.83 e la deviazione standard, che indica la dispersione dei dati rispetto alla media, è uguale a 88.94[4]. Da questi dati statistici si comprende che il dataset fornito è non IID. Infatti, se la deviazione standard è maggiore di 0 significa che i dataset singoli di ogni utente non sono uguali, ma possono avere un numero diverso di immagini nel training-set e nel test-set. Questo aspetto è confermato dall'articolo che presenta il benchmark nel quale viene affermato che i punti dati sono distorti tra i dispositivi[4].

Tenendo conto di questa caratteristica peculiare del dataset FEMNIST, è fondamentale considerarla sia per l'implementazione del progetto sia per l'analisi dei risultati dell'addestramento dei modelli basati su tale dataset.

2.3 Da MNIST a FEMNIST

In principio per l'ambiente del FLI veniva utilizzato MNIST, tentando però di simulare l'adattamento a sistemi federati che era proprio FEMNIST.
In quel caso il dataset veniva partizionato programmaticamente, dividendo semplicemente l'intero set di addestramento e set di test in 100 parti, tanti quanti sarebbero stati i partecipanti alla simulazione. Per il partizionamento in 100 set di MNIST veniva utilizzata la funzione di PyTorch `random_split()`[24]. Questa faceva in modo che le immagini nel database venissero prese casualmente dal totale sia per il set di addestramento che per quello di test, e infine si ottenevano insiemi di dati con un identico numero di campioni.
Questa configurazione aveva la possibilità di essere non-IID, poiché la distribuzione degli esempi di ogni classe era casuale. Tuttavia, la questione importante era che, siccome veniva impiegato tutto il dataset MNIST, le prestazioni del modello erano inverosimilmente alte, tanto che l'accuratezza arrivava al 99,7% addestrando con questi dati il modello tramite il FL.
Apparentemente questa elevata accuratezza può sembrare un aspetto positivo del modello ma in realtà essa è la causa dell'overfitting. L'overfitting è quel fenomeno che accade quando un modello non si generalizza bene dai dati osservati ai dati nuovi, mai incontrati prima. A causa dell'overfitting, il modello funziona perfettamente sul set di addestramento, mentre si adatta male al set di test. Inoltre, i modelli che presentano overfitting tendono a memorizzare tutti i dati, compreso l'inevitabile rumore presente nel train-set, invece di imparare lo schema nascosto dietro i dati[48].
La soluzione all'elevata accuratezza e al conseguente overfitting è stata utilizzare FEMNIST, considerando anche che è specificatamente progettato per ambienti federati. Esso infatti è fornito già partizionato e, oltretutto, delle migliaia di partizioni, per il progetto si necessitava di solamente 100 di queste, tante quante sono i client che partecipano al

FL. A questo punto l'apprendimento, usufruendo di una quantità di dati ridotta, avrebbe portato a risultati più adeguati e che non soffrissero di overfitting.
Usufruire di FEMNIST invece che di MNIST partizionato programmaticamente, in questo contesto, oltre che come rimedio alla troppo alta accuratezza del modello, è stato un valore aggiunto per il sistema ibrido portando molti vantaggi:

- **Rappresentazione realistica dei dati**
  FEMNIST è stato creato utilizzando i dati originali del NIST che sono stati rielaborati in modo tale che questi siano strettamente legati allo scrittore originale dei numeri e caratteri, cioè un client di FEMNIST corrisponde esattamente ad uno scrittore che ha partecipato alla raccolta dei dati per NIST. Pertanto, poiché ogni scrittore ha uno stile e calligrafia unica, questo datatset esibisce il tipo di comportamento non-IID che ci si aspetta dai dataset federati[16].

- **Benchmark per il FL**
  FEMNIST è stato creato con lo scopo di fornire un benchmark gli ambienti federati e quindi essere un punto di riferimento. Questo significa che è possibile confrontare i risultati ottenuti con quelli di altri ricercatori che utilizzano questo dataset.

- **Dataset già partizionato**
  In quanto FEMNIST è fornito già diviso per client, non c'è necessità di farlo all'interno del programma, ma basta solamente attingere al dataset specifico di ogni client[4].

2.4 Il dataset *FEMNIST_pytorch*

Il dataset FEMNIST fornito da LEAF è progettato per funzionare con un ecosistema TensorFlow e Numpy[20, 28], mentre il progetto del sistema ibrido è basato su un ambiente PyTorch. Perciò, per l'implementazioni è stata utilizzata la repository su GitHub[17] *FEMNIST_pytorch*, il cui autore è tao-shen (`https://github.com/tao-shen/FEMNIST_pytorch`)[35], che forniva FEMNIST con supporto per PyTorch. Questo è l'url dal quale scaricare il dataset `https://raw.githubusercontent.com/tao-shen/`

`FEMNIST_pytorch/master/femnist.tar.gz`. Il dataset ottenuto, come specificato dall'autore stesso, deriva direttamente dalla repository ufficiale di LEAF[21]. È importante soffermarsi sulle differenze tra FEMNIST ottenuto dalla repository *FEMNIST_pytorch* e quello ufficiale. La prima differenza è che esso è composto da sole 10 classi e non dalle 62 classi, ciò fa presumere che questa versione del dataset è una versione federata delle sole immagini di cifre scritte a mano, senza includere le lettere dell'alfabeto, maiuscole e minuscole. D'altronde questa divergenza si riflette nel numero di campioni totali che equivale a 341,873 per il set di addestramento e 40,832 per il set di test, con il totale dei client che è pari 3383.
Dividendo i campioni per il numero di utenti si ottiene una media di circa 101 immagini per il train-set e circa 12 per il test-set. Controllando gli esempi assegnati singolarmente ad ogni client si riscontrava che effettivamente alcuni ne avevano di più, altri di meno. Per implementare l'ambiente di ibridazione del FL con il ML centralizzato si necessitava di soli 100 utenti partecipanti e tra tutti quelli presenti sono stati selezionati i primi.
Osservando i numeri di campioni di questa versione del dataset e confrontandoli con quelli di FEMNIST fornito da LEAF è chiara la differenza per quanto riguarda le immagini presenti nel dataset, ma non si può dedurre nulla riguardo alle classi.
A questo punto però sono sorti dei dubbi sull'affidabilità del dataset *FEMNIST_pytorch*, anche perché, né nella documentazione della repository, né nel codice fornito, si faceva riferimento con certezza al fatto che questo dataset era una versione di FEMNIST che conteneva solo e soltanto le cifre e che, per contro, fosse un semplice sottoinsieme dell'originale FEMNIST.
Era necessario quindi verificare la veridicità delle supposizioni relative alla struttura del dataset. Affinché potesse essere eseguito questo accertamento era necessario controllare che il set di dati contenesse realmente solo immagini di cifre e per fare ciò bisoganva prelevarne una parte e analizzarle. Però, quando si parla di immagini nel contesto concreto dell'implementazione del dataset, in realtà si fa riferimento a matrici. Infatti una volta scaricato ed estratto il train-set e il test-set si ottengono array bidimensionali 28x28, che rappresentano immagini in scala di grigi, contenenti numeri float tra 0 a 1. Di conseguenza, comprendere da una matrice così strutturata e che contiene tali valori, se questa rappresenti una lettera o un numero è inverosimile. Tuttavia, ammettendo che questa

verifica fosse fattibile, si consideri che era necessario cercare all'interno del dataset quel campione che smentisse l'ipotesi, cioè che questa configurazione di FEMNIST contenga solo e soltanto immagini di cifre scritte a mano.
In questo articolo «An Experimental Study of Class Imbalance in Federated Learning», come afferma il titolo, viene presentato uno studio sperimentale sullo sbilanciamento delle classi in apprendimento federato, che prende in considerazione tre dataset tra cui anche FEMNIST. Proprio riguardo a quest'ultimo l'articolo dichiara che *"FEMNIST is a federated version of MNIST dataset with 341873 samples from 3383 writers which is clients in federated learning"*[45] e successivamente *"Each client in original FEMNIST dataset contains around 100 samples from 10 classes"*[45]. Precisa dunque che il datatset FEMNIST che viene adoperato ha 341873 campioni da 3383 scrittori, cioè client, e ognuno di questi contiene circa 100 campioni da 10 classi.
Si nota quindi che il dataset utilizzato in questo studio ha esattamente lo stesso identico numero di campioni e utenti di quello scelto per l'implementazione del FLI, di conseguenza si può confermare l'ipotesi, la quale affermava che tale configurazione del dataset contiene le solo e soltanto 10 classi, corrispondenti alle cifre.
In aggiunta lo studio affronta l'argomento dello squilibrio delle classi all'interno di un ambiente federato, che non è da trattare nello stesso modo del ML centralizzato. Viene definito lo squilibrio di classe a livello globale e locale, poiché una classe può essere maggioritaria all'interno del dataset di determinati client, ma allo stesso tempo potrebbe essere una classe minoritaria del dataset a livello globale. I due ambiti non sono in relazione. Le conseguenze di uno squilibrio, ad entrambi i livelli, si riflette sulle prestazioni del modello globale, che degradano con l'aumentare delle disparità. In particolare, viene mostrato come aumentando la dimensione del dataset aumenti in modo direttamente proporzionale il livello di distorsione: se i dati nel set raddoppiano, lo squilibrio delle classi raddoppia. Inoltre, un alto livello di squilibrio locale delle classi è stato dimostrato rallentare la convergenza del modello globale, sviando l'ottimizzazione.
Nello studio riguardo a FEMNIST e ciò che concerne la distorsione delle classi viene affermato che esso a livello globale è quasi bilanciato tra le classi, quindi non ci sono classi considerate spiccatamente maggioritarie o minoritarie. Mentre, invece, esiste uno squilibrio locale tra i client a causa di eccezioni e casi anomali[45].

Le considerazioni da fare a questo punto sono due:

- **Ridotta dimensione del dataset**
  Dunque, se aumenta la dimensione del set di dati aumenta proporzionalmente anche lo squilibrio. La proporzionalità diretta vale anche nella direzione opposta, perciò, nel caso specifico della ricerca sul FLI, avendo un dataset molto esiguo in quanto viene impiegato solo il 2.93% della sua dimensione totale, anche lo sbilanciamento sarà ridotto.
  Comunque anche lo studio afferma che FEMNIST di base ha un buon grado di equilibrio globale tra le classi[45].

- **Ridotto numero di client**
  Il numero di client presi in considerazione nel progetto è molto limitato, solo 100 su 3383, che equivale a circa il 3% dei totali, pertanto, diminuisce di molto anche la possibilità di scegliere tra quei 100 utenti dei casi anomali evidenziati dallo studio[45].

Date tutte le precedenti considerazioni, si può affermare che la stuttura del dataset sia inequivocabile, esso è composto da sole 10 classi con uno sbilanciamento globale e locale che, anche se presente, è comunque di entità così ridotta da poter essere trascurabile.

2.4.1 Utilizzo del dataset

Questa sezione si concentrerà su come sia stato personalizzato l'utilizzo del dataset FEMNIST per adattarlo alle esigenze specifiche del sistema soggetto di questo elaborato. I client che partecipano all'apprendimento in questo ambiente sono costanti e pari a 100. FEMNIST fornisce 3383 utenti, indicando per ognuno il numero di campioni che compongono il set privato di ogni partecipante, sia per il train-set sia per il test-set. Pertanto, il totale delle 341873 immagini che compongono FEMNIST sono partizionate nei 3383 dataset locali. La classe `FEMNIST` stessa indica, nel parametro `users_index`[35], quanti campioni appartengono ad uno specifico client. Quindi, a partire dall'utente con indice 0, vanno assegnati n campioni in base a quelli indicati nel parametro `users_index[`x`]`, con x che è l'indice dell'utente che va da 0 a 99.

Una volta eseguita questa operazione si ottengono 100 utenti, ognuno dei quali ha assegnato un numero diverso di campioni. I campioni totali di tutti i set di addestramento sono 9432, ciò vuol dire che ogni client possiede in media circa 94 campioni, mentre per il set di test il totale è pari a 1134, con una media per dataset locale di circa 11 esempi. Per avere un idea della misura della porzione di dataset utilizzato per l'implementazione rispetto alla totalità dei dati si può mettere in relazione le quantità di campioni. Per il set di addestramento abbiamo che il numero totale di campioni è 341,873, di cui solo 9432 di questi compone il dataset del progetto, che corrisponde al 2.76% del totale. Per quanto riguarda il test set la porzione presa in considerazione è pari al 2.78%, cioè 1134 su 40382. Una considerazione conclusiva sull'argomento del dataset in questione è che per il progetto viene utilizzato un sottoinsieme di FEMNIST che include i dataset locali dei primi 100 utenti specificati, porzione corrispondente al 2.76% del totale.

Capitolo 3

Implementazione

In questo capitolo dell'elaborato verrà affrontato e articolato integralmente il concetto di Federated Learning Ibrido (FLI) o Hybrid Federated Learning. In alcuni articoli si parla di Hybrid Federated Learning[49, 51], ma non in questo senso, intendendo invece il Federated Transfer Learning, in quanto ibrido tra il FL orizzontale e il FL verticale. Tuttavia, in questa ricerca ci si riferisce al FLI come al principio di combinare e far coesistere, ibridare appunto, il FL con il ML centralizzato nello stesso sistema. L'idea è emersa immaginando un caso d'uso in cui in un ambiente federato nel quale valgono i principi di riservatezza dei dati, che vengono mantenuti localmente, e apprendimento decentralizzato, cioè che avviene all'interno del client, alcuni di questi utenti al contrario siano disposti a partecipare all'apprendimento automatico in modo centralizzato, rinunciando perciò ai principi su cui si basa il FL.

3.1 L'idea dell'ibridazione

Questa concezione lascia spazio alla nascita di un nuovo sistema in cui i client che partecipano all'addestramento sono liberi di fare una scelta tra due approcci di apprendimento automatico:

- **Modalità federata**
  I client che scelgono questo approccio mantengono tutti i diritti che vengono preservati nel Federated Learning classico e si avvalgono dei principi su cui esso è

fondato. Perciò, vale la privacy dei dati, i quali rimangono memorizzati localmente all'interno del dispositivo del client; la fase di apprendimento del modello rimane decentralizzata, di conseguenza l'addestramento viene eseguito nel dispositivo; la comunicazione con il server centrale e la condivisione di dati avviene solamente per inviare i parametri del modello allenato e ricevere il modello globale aggregato.
Insomma, i client che optano per questo metodo si comportano come si comporterebbero in un ambiente federato.

- **Modalità centralizzata**
  I client che decidono di adottare l'approccio centralizzato, invece, partecipano all'apprendimento in maniera completamente opposta alla precedente. Ciò implica che: si accetta di inviare i propri dati al server centrale che li memorizzerà in un database comune; l'apprendimento viene eseguito pertanto dal server centrale, con tutti i dati memorizzati nel suo database provenienti da tutti i client che hanno scelto questo approccio; la comunicazione con il server si limita a due scambi che si collocano agli estremi temporali dell'intera esecuzione dell'apprendimento, cioè all'inizio, in cui il client in questione invia i propri dati al server, e alla fine, in cui il server invia il modello globale addestrato.
  Fondamentalmente i client che prendono la decisione di adottare questo metodo si comportano come utenti di un sistema di ML centralizzato.

Si rivela questa un'idea innovativa, non perché viene presentata una nuova tecnologia, ma perché viene proposto un inedito e rivoluzionario ambiente di apprendimento automatico. In questo nuovo sistema non solo coesistono ma collaborano due tecniche opposte di machine learning per un obbiettivo comune: l'allenamento di un modello condiviso. Tutti i client che faranno parte dell'ambiente ibrido ne usufruiranno, a prescindere dall'approccio scelto. Pertanto in questo contesto inedito è possibile beneficiare dei vantaggi di entrambe le tecniche di apprendimento e, al tempo stesso, smussarne i rispettivi limiti.

3.2 Implementazione concettuale

Verrà ora concettualizzata l'implementazione del FLI, cioè sotto quale forma e con quali mezzi questo sistema ibrido è stato realizzato. Innanzitutto la base sulla quale

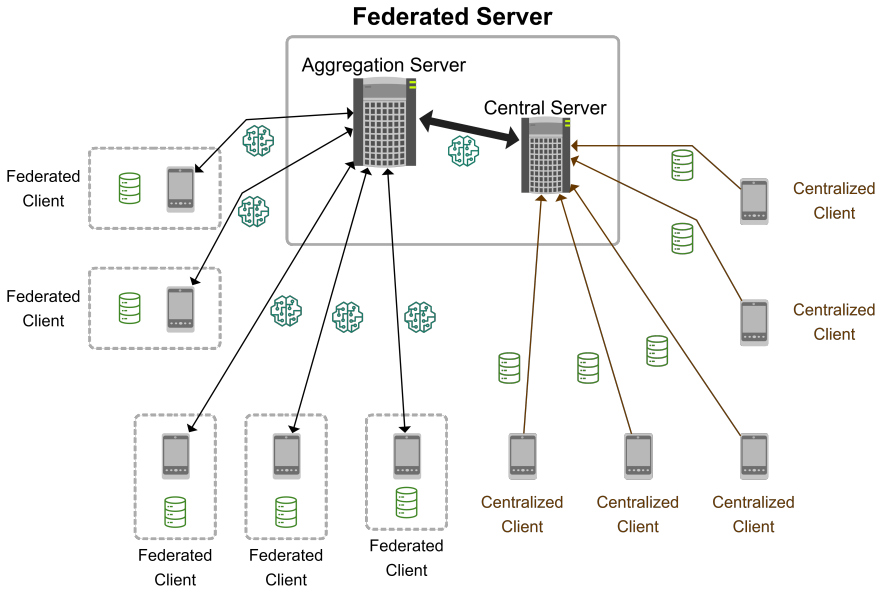

Figura 3.1: Schema rappresentativo del Federated Learning Ibrido

costruire il sistema è il FL, poiché esso tra le due tecniche è quello più restrittivo, quindi risulta più agevole ideare un modo per applicare un sistema centralizzato ad un ambiente federato.

Per verificare la concretezza di questa ipotesi basta assumere di voler procedere nel modo opposto, quindi implementare il FL all'interno di un sistema centralizzato. Questa direzione si rivela estremamente più laboriosa e complessa. Infatti sarebbe stato necessario costituire tutta l'architettura che permettesse di: avere una comunicazione costante tra server e client in ogni round per condividere solo e soltanto i modelli; integrare nel server il metodo di aggregazione dei modelli e delle metriche provenienti dai client federati; assicurare che la privacy dei dati dei client federati venisse rispettata; permettere ai client che adottavano la modalità decentralizzata di eseguire per conto proprio sia l'apprendimento che la valutazione del modello. Dunque, ci si rende conto che seguire questa direzione progettuale per realizzare il FLI, volendo implementare tutte le specifiche del FL all'interno di un progetto di apprendimento centralizzato, era operativamente troppo gravoso e probabilmente anche inefficiente.

Risultava necessario, a tal proposito, trovare il modo di mettere in atto questo sistema

duale, nel quale in un ambiente di FL convivessero client che volessero partecipare all'apprendimento in modalità centralizzata con client decentralizzati. L'intuizione geniale è stata quella di simulare questo comportamento di client centralizzati e server tradizionale, senza aggiungere complessità all'ambito federato o modificarlo, ma semplicemente implementarlo all'interno di un unico client che partecipasse al FL.
In sostanza, il funzionamento sarebbe stato questo: gli utenti che sceglievano l'approccio centralizzato venivano accorpati tutti quanti in un solo client che avrebbe partecipato all'apprendimento federato, più specificatamente erano i loro dati che venivano uniti in uno stesso set. L'aggregazione di questi dataset diveniva il set locale di un singolo client, che, a questo punto, si comportava come un qualsiasi altro dispositivo che aveva optato per la modalità federata. In questo modo tale client "centralizzato" avrebbe assunto tutte le funzioni che il server non federato avrebbe dovuto svolgere. In effetti, esso possiede tutti i dati dei client in modalità tradizionale, esegue l'apprendimento con il dataset composto dall'aggregazione di tutti i campioni di quei client e, di conseguenza, quei client si svuotano di tutte le funzioni operative condividendo i propri dati. Ciò equivale quindi a implementare la centralizzazione del processo, invece che all'interno del server, in un modulo esterno ad esso.

3.3 Implementazione in Python

Dopo aver inquadrato il concetto di Hybrid Federated Learning presentato in questa ricerca, è necessario affrontare come operativamente è stato implementato l'ambiente ibrido in questione.

3.3.1 Python e ambiente di sviluppo

Innanzitutto, il linguaggio di programmazione, che è stata quasi una scelta obbligatoria: Python[8]. Scelta obbligatoria perché tutti i framework e librerie principali per creare un ambiente di apprendimento automatico sono sviluppate in Python. Ma quindi perché è il linguaggio più utilizzato quando si parla di machine learning, intelligenza

artificiale, data analysis e deep learning. Vediamo brevemente le motivazioni principali del suo sviluppo in questa direzione:

- La sintassi è stata progettata per essere semplice e intuitiva, il che lo rende un linguaggio di facile lettura. La programmazione è orientata agli oggetti. Queste caratteristiche e la sua facilità di utilizzo lo hanno reso un linguaggio accessibile per i principianti e, allo stesso tempo, semplice nello sviluppo e debug per i più esperti.

- Un fattore chiave è il suo completo e articolato ecosistema di librerie e framework progettate specificatamente per l'apprendimento automatico come: NumPy, PyTorch, Tensorflow.

- La compatibilità multi-piattaforma garantisce flessibilità per gli sviluppatori, ma bisogna far attenzione a non incorrere in problemi di compatibilità per via delle diverse versioni di Python.

- La sua scalabilità e le sue eccezionali prestazioni gli conferiscono ampio prestigio nel mondo del machine learning. Comunque, nonostante sia un linguaggio interpretato ci sono anche librerie e framework che ne migliorano notevolmente le prestazioni[36].

- L'ultimo aspetto non riguarda Python in sé ma la community di sviluppatori che c'è intorno a questo linguaggio. Community molto attiva e che può fornire supporto e aiuto[27].

Era necessario allora individuare l'ambiente di sviluppo in cui implementare il progetto in Python. Dapprima è stato adoperato Visual Studio Code (VS Code), un editor di codice sorgente molto leggero ma allo stesso potente e personalizzabile grazie a tutte le estensioni che si possono integrare[44]. Insieme a VS Code, è stato utilizzato Conda, gestore di pacchetti e dell'ambiente open source Python, per installare i pacchetti necessari e gestire l'ambiente virtuale[39]. Quando, dopo aver sviluppato in parte il sistema, questo è iniziato a diventare troppo pesante, notando che necessitava di più risorse computazionali, si è abbandonata l'idea di implementare tutta l'architettura ed eseguire la computazione in locale.

La risoluzione di questi problemi è stata trovata in Google Colaboratory[15], noto anche come Colab in breve, che è un prodotto di Google Research. Colab consente a chiunque di eseguire codice python attraverso il browser ed è particolarmente adatto per l'apprendimento automatico e l'analisi dei dati. Esso tecnicamente è un servizio di hosting di notebook Jupyter che non richiede alcuna configurazione e fornisce accesso gratuito alle risorse di calcolo, comprese le GPU[14]. I Jupyter Notebook sono semplicemente dei documenti interattivi nei quali è possibile scrivere ed eseguire del codice, in questo caso python. Questi documenti possono essere divisi in moduli a sé stanti, con la possibilità di interscambiare sezioni di codice e testo informativo, formattato in Markdown[11]. Il grande vantaggio di un servizio di hosting del genere è esattamente quello di fornire risorse di calcolo virtuali, e quindi eseguire il proprio codice in cloud. Non importa quali caratteristiche e risorse abbia il dispositivo sul quale si usa, basta solo connessione ad Internet e un account Google[14]. Grazie a Colab quindi è possibile eseguire operazioni computazionalmente dispendiose in cloud, liberando la propria macchina.

3.3.2 Framework e librerie

I framework e le librerie di Python alla base dell'implementazione del Federated Learning e di questo sistema del FLI in particolare sono: Flower[2] e PyTorch[8].
È particolarmente rilevante in questo contesto approfondire Flower, che risulta un game-changer nell'ambiente federato per quel che riguarda strumenti e potenzialità. Flower è un nuovo framework di apprendimento federato end-to-end che consente una transizione più fluida dalla ricerca sperimentale in simulazione alla ricerca di sistema su un'ampia gamma di edge device reali. Flower offre la possibilità alle implementazioni sperimentali di migrare tra i due estremi secondo le necessità. I suoi obiettivi di progettazione si propongono di colmare delle lacune lasciate dagli esistenti ecosistemi di FL e sono:

- La scalabilità per supportare un gran numero di client simultanei per promuovere la ricerca su una scala realistica. Questo framework arriva a supportare 1000 client concorrenti e un milione di client totali.

- Essere indipendente dai client, in particolare i dispositivi mobili, per adattarsi su ognuno e garantire interoperabilità.
- Avere un'indipendenza anche dalla comunicazione date le differenze di connettività dei dispositivi che potrebbero partecipare.
- Essere indipendente anche dalla privacy poiché configurazioni di sistemi FL diversi hanno requisiti di privacy diversi.
- Risultare flessibile data la velocità di aggiornamento dell'ecosistema dell'apprendimento federato e del ML in generale.

Tutti questi requisiti di progettazione hanno dato vita ad uno strumento molto efficace, estremamente adattabile e malleabile e che non dipende da linguaggi di programmazione e da framework di ML[3]. Visto che questo progetto aveva come scopo la ricerca e l'analisi di un sistema non reale ma ideale mi sono avvalso dell'implementazione di una simulazione. Flower come già detto permette di simulare questo sistema di FLI per poterne valutare le prestazioni e analizzarne i risultati senza la necessità di dover testare e implementare il tutto in un contesto concreto composto da dispositivi edge reali.
Flower è progettato, come abbiamo visto, per funzionare con librerie specifiche per l'apprendimento automatico a piacimento e nel caso di questo progetto la scelta è ricaduta su PyTorch, una libreria per programmi Python che facilita la costruzione di sistemi basati sul Deep Learning (DL). Il DL si può semplificare concettualmente ed esprimerlo sotto forma di una complessa funzione matematica che mappa gli input in output. Per facilitare l'espressione di questa funzione PyTorch fornisce una struttura dati di base, il tensore, che è un array multidimensionale paragonabile ad un array di Numpy[34, 28]. Proprio i tensori saranno la struttura dati che costituiranno il dataset locale dei vari client che parteciperanno al FL. Questi dataset sono composti da quasi un centinaio di immagini, che in PyTorch verranno tradotte in tensori tridimensionali composti da matrici 28x28 di valori tra 0 e 1, in quantità diversa in base al client. Queste matrici sono l'espressione matematica delle immagini in scala di grigi di cifre scritte manualmente costituenti FEMNIST.
Questa libreria per il DL fornisce oltre al tensore anche operazioni su di esso nel modulo `torch`; contiene un modulo `torch.nn` con tutte le funzioni necessarie per creare e gestire

le reti neurali, incapsulandole in classi e trattandone i parametri sotto forma di variabile; tramite il modulo `torch.utils.data` si dà la possibilità allo sviluppatore di gestire i dati e i dataset tramite classi predefinite e operazioni su di esse[34].
Flower e PyTorch si sono rivelati degli strumenti utilissimi ed estremamente precisi nelle funzionalità. Tutto ciò di cui si necessitava riguardo al processo federato, alla gestione dei client, della strategia e del server era fornito da Flower. Qualsiasi oggetto o funzione riguardasse le reti neurali, il modello, i parametri, la gestione del dataset e le operazioni di addestramento e test del modello veniva governato da PyTorch.

3.3.3 Procedimento operativo del sistema

Verrà analizzato dunque il processo che segue questo sistema del FLI, che in sostanza si tratta di una simulazione di una rete di client diversi che partecipano ad un addestramento federato, di cui uno è quello centralizzato.
Il procedimento parte dal dataset *FEMNIST_pytorch*. Train-set e test-set vengono recuperati dall'apposita classe `FEMNIST`, fornita dalla repository appena citata[35], che in parte ho modificato per adattare alle necessità di questo progetto. Se i dataset sono stati scaricati precedentemente, si recuperano tutti i dati forniti dal dataset FEMNIST, altrimenti lo si scarica dalla fonte indicata nella repository (`https://raw.githubusercontent.com/tao-shen/FEMNIST_pytorch/master/femnist.tar.gz`). Ottenuti i set di addestramento e di test (d'ora in poi quando ci si riferisce a set si intende sia train-set che test-set) vengono selezionati i primi 100 client, di conseguenza si creano 100 `Subset`[24] dai set originali, ognuno dei quali contiene il numero di campioni, a partire dall'inizio del set, specificato nel parametro `users_index` della classe `FEMNIST`. Questo è il codice in cui vengono prelevati i 100 client dal dataset totale:

```
for i in range(num_partitions):
    num_images = dataset.users_index[i]
    datasets.append( Subset(
        dataset,
        range(images_taken, images_taken + num_images))
    )
    images_taken += num_images
```

dove `datasets` equivale all'insieme dei set. A questo punto avremmo i singoli dataset da assegnare ognuno ad un client.
Successivamente si posiziona la fase in cui viene applicata la personalizzazione al sistema federato. Pertanto, vengono gestiti i vari client in base al numero di questi che affronteranno l'apprendimento in modalità federata o centralizzata. Se tutti i client sono federati si prosegue nella preparazione dei dataset per la simulazione e quest'operazione viene saltata, altrimenti, si procede con l'unione dei set dei client centralizzati, il cui numero sarà espresso dalla costante X. La funzione del progetto in cui avviene questa operazione è la seguente:

```
def merge_datasets(trainsets, testsets):

    merged_trainsets = []
    merged_testset = []
    if len(trainsets) == len(testsets):
        rand_indexes = random.sample(range(0, len(trainsets)), X)
        rand_indexes.sort()
    else:
        raise Exception("trainsets and testsets must have the same length")

    for i in rand_indexes:
        merged_trainsets.append(trainsets[i])
        trainsets[i] = None
        merged_testset.append(testsets[i])
        testsets[i] = None

    trainsets = [x for x in trainsets if x is not None]
    testsets = [x for x in testsets if x is not None]
    trainsets.append(ConcatDataset(merged_trainsets))
    testsets.append(ConcatDataset(merged_testset))

    return trainsets, testsets
```

Il merge dei dati avviene prelevando in modo casuale X indici sul numero dei client totali. Successivamente quei dataset scelti randomicamente verranno accorpati in un unica struttura dati, fornita da PyTorch dalla classe `ConcatDataset`. Questa classe sostanzialmente permette di concatenare e incapsulare dataset distinti all'interno di una sola struttura dati, che verrà trattata come se fosse un singolo dataset[24]. Il set originale, che intanto è stato privato dei client centralizzati, avrà una dimensione pari a alla lunghezza originale meno gli X set prelevati, e a questa struttura risultante viene aggiunto in coda il set `ConcatDataset`, contenente l'unione dei set dei client centralizzati. La dimensione finale della lista di dataset è pari a $N - X + 1$, con N che è il numero di client totale. Ciò che avremo in questo momento è un numero di set pari al numero di client federati più uno, che rappresenta il client che gestisce il modulo centralizzato.
In seguito si procede con la generazione dei set di convalida a partire dai set di addestramento, che avviene prelevando da quest'ultimo il 10% dei dati. Il rapporto tra train-set e validation-set è sempre pari a 9:1. Per questa operazione viene utilizzata la funzione di PyTorch `random_split()`, al quale viene passato il train-set di un client, la dimensione delle parti in cui il dataset deve essere scisso e un generatore utilizzato per la permutazione casuale dei dati. A questo generatore viene associato un seed, o seme, che permette la riproducibilità dell'operazione, infatti, usando lo stesso seme si ottiene sempre la stessa permutazione. Questo consente di avere un controllo consistente sull'ambiente sperimentale, al fine di ottenere dei risultati validi. Il codice successivo è eseguito all'interno di un ciclo `for` che scorre tutti i trainset:

```
for_train, for_val = random_split(
    trainset_,
    [num_train, num_val], generator=torch.Generator().manual_seed(2023)
)
```

L'ultima operazione riguardante la preparazione del dataset è quella di ottenere un `DataLoader` per ogni singolo set di addestramento, di convalida e di test. La classe `DataLoader` è essenziale per l'elaborazione dei dati, essa combina un dataset e un campionatore e fornisce un oggetto iterabile sul set di dati passato per argomento. Il suo scopo principale è quello di supportare il caricamento automatico dei dati in batch. Per batch si intende un lotto, una piccola porzione, dei dati totali che compongono un data-

set. Questo aspetto è particolarmente sfruttato per l'addestramento di reti neurali, nel quale il training viene eseguito su un batch di dati alla volta. In più `DataLoader` permette di fare uno shuffling dei dati, utile quando si tratta del training-set, per migliorare la generalizzazione e far concentrare l'addestramento del modello sui singoli dati e non sulla loro sequenza[24].
Questi DataLoaders saranno poi passati ai client che se ne serviranno nelle apposite fasi dei round. Per essere precisi, in realtà, nella simulazione di un apprendimento federato con Flower, ad ogni operazione che coinvolge i client, il server si servirà di una funzione per generarli all'occorrenza ogni volta. Il server, a tal proposito, si comporta come il regista della simulazione, di fatto, ogni volta che deve essere eseguita la fase di apprendimento o quella di valutazione, esso crea i singoli client e gli fornisce il modello globale e il numero del round attuale. I client da parte loro ritornano dopo la fase di train: i parametri del modello, il numero di campioni del proprio dataset e le metriche; per la fase di test ritornano le metriche risultanti e il numero di campioni del set utilizzato in questa fase[2].
La strategia implementata dal server federato è stata la più comune per il FL cioè FederatedAveraging (`FedAvg`).
Quindi bastava fa partire l'esecuzione della simulazione e attendere la fine per raccoglierne i risultati. Pertanto, per concludere la discussione sull'implementazione del FLI, questa si può riassumere affermando che viene simulato un sistema di FL dove i client federati si comportano come in un normalissimo FL. I client in modalità centralizzata, invece, vengono uniti insieme per generare un solo e unico client, che però si comporterà esattamente come tutti gli altri client federati. Allo stesso tempo però sarà come se esso rappresentasse il server centrale che riceve i dati dai client e si occupa di apprendimento e valutazione del modello.

Capitolo 4

Valutazione

Questo capitolo rappresenta il culmine dello studio sul Federated Learning Ibrido, poiché verranno esaminati in dettaglio i dati raccolti dalle simulazioni e da questi verranno tratte delle conclusioni significative.
Si inizia con una descrizione dell'ambiente di ricerca, come sono stati prodotti i risultati delle simulazioni e quali sono le altre tecniche di ML di cui ci siamo serviti come punti di riferimento. Successivamente si avrà una panoramica generale dei risultati, esaminando le tendenze emergenti e confrontando le prestazioni del FLI con gli altri metodi di apprendimento automatico. Verrà poi discussa l'importanza di questi risultati nel contesto più ampio del campo dell'apprendimento automatico e consideriamo le potenziali implicazioni per future ricerche. Infine, saranno presentati dei punti di riferimento per poter migliorare questo sistema ibrido.
Questo capitolo non solo fornisce una conclusione al nostro lavoro, ma pone anche le basi per ulteriori indagini ed applicazioni in questo affascinante e in rapida evoluzione campo di studio.

4.1 Ambiente di ricerca

Il primo argomento da affrontare è l'ambiente di ricerca in cui questo studio è avvenuto. Con ambiente di ricerca si intende tutto il contorno, le caratteristiche e i valori che sono stati considerati appunto perché si tratta di una ricerca, non di un'applicazione del

sistema ad un contesto reale. Ovviamente tenendo in cosiderazione che uno studio deve il più possibile simulare un ambiente reale poiché dopotutto è questo il suo scopo. Per fare un esempio: in un contesto reale i client avrebbero avuto il libero arbitrio sulla scelta di quale delle due modalità adottare all'interno di un apprendimento con FLI, mentre, in questo studio il numero di client che sceglievano la modalità centralizzata erano decisi a priori, al fine della ricerca e di esplorare il più possibile il potenziale di questo innovativo sistema ibrido.
Le caratteristiche dell'ambiente di ricerca sull'Hybrid FL sono le seguenti:

- **Client e dataset**
  Il numero di client che partecipavano ad ogni simulazione erano sempre 100 e i dataset relativi ad ognuno rappresentavano i primi 100 che venivano indicati nell'array `user_index` della classe `FEMNIST`. Unire alcuni di questi dataset insieme in un singolo insieme aveva lo stesso significato di inviare questi dati ad un server centrale con lo scopo di memorizzarli.
  In realtà, in base alla configurazione e alla tecnica messa in atto il numero di client che concretamente partecipavano alla simulazione potevano essere meno di 100, ma i dati che venivano elaborati erano sempre gli stessi e in numero sempre constante, ed era questo che rendeva consistente la configurazione.

- **Livello di ibridazione**
  Il grado di ibridazione del sistema era variabile e sono stati testati diversi livelli. Tutto dipendeva dalla variabile X, che all'interno della simulazione singola diveniva una costante. Essa esprimeva il numero di client che adottavano la modalità centralizzata. I valori assunti da X erano 0, 20, 40, 60, 80, 100. $X = 0$ equivaleva al Federated Learning classico e $X = 100$ rappresentava un apprendimento totalmente centralizzato. In linea con i valori assunti da X diversi livelli di ibridazione erano 0%, 20%, 60%, 80%, 100%, oppure possiamo esprimere le tecniche con la scala invertita, cioè FL al 100% (ibridazione = 0%), FL all'80% (ibridazione = 20%) fino al FL allo 0% che corrisponde al Centralized Learning

- **Tecniche di ML**
  Le tecniche di apprendimento automatico utilizzate come punti di riferimento, oltre

al FL completo, le versioni ibride e la versione centralizzata, comprendevano anche il Single Learning. Esso rappresenta una personalizzazione del FL classico nel quale ognuno dei 100 client opera in singolo, senza comunicare con il server nemmeno i propri parametri dopo le fasi di addestramento. In questa tecnica ogni client addestra in solitario il proprio modello, non c'è alcuna aggregazione e condivisione di un modello globale. Solamente le metriche misurate vengono ricevute dal server per fornire i valori totali di tutta la rete.

- **Simulazioni**
  Per avere dei risultati consistenti sui quali effettuare un analisi affidabile le simulazioni, per ogni singola tecnica di apprendimento e livello di ibridazione, sono state ripetute almeno 5 volte, sia con 3 che con 5 round di apprendimento. I risultati quindi si basano su un minimo di 70 simulazioni totali.
  La configurazione di ogni simulazione di FL, a parte il numero di round, è rimasta fissa. Come abbiamo già detto il numero di dataset era sempre 100 e il numero di client della simulazione in Flower risultavano diversi in base al livello di ibridazione, da 100 del FL a soltanto 1 della modalità totalmente centralizzata. Altri valori di configurazione sono quelli riguardanti l'addestramento: learning rate pari 0.0004, momentum uguale a 0.9 e le epochs locali, cioè le ripetizioni di ogni addestramento all'interno di ogni client, erano sempre una.

- **Metriche misurate** I valori recuperati da ogni simulazione, sui quali è stata basata l'analisi, sono 3:

  1. *Tempo di esecuzione*
     Il numero di secondi della durata di ogni singola simulazione

  2. *Accuracy*
     L'accuratezza totale del modello, ottenuta aggregando tutte le accuracy dei modelli di ogni client. È calcolata valutando il modello locale con il proprio set di test ed è utilizzata per valutare i modelli di classificazione. Informalmente, è la percentuale di previsioni che il nostro modello ha ottenuto correttamente[13].

3. *F-score*
   È la misura dell'accuratezza di un test, aggregando, anche in questo caso, le misure di ogni client. Chiamato anche F1 score o F-measure, viene calcolato a partire dalla precisione e dal recall del test, dove la precisione è il numero di risultati positivi veri diviso per il numero di tutti i risultati positivi, compresi quelli non identificati correttamente, e il recall è il numero di risultati positivi veri diviso per il numero di tutti i campioni che avrebbero dovuto essere identificati come positivi. L'F1 score è la media armonica della precisione e del recall[40].

Apparentemente accuracy e f-score possono sembrare uguali, ma in realtà sono necessari entrambi per la valutazione di un modello. Le differenze chiave tra le due sono le prestazioni su set di dati sbilanciati: l'f-score è in grado di misurare le prestazioni in modo oggettivo quando il bilanciamento delle classi è distorto; mentre l'accuracy è sì universalmente compresa, ma in presenza di dati troppo sbilanciati non si comporta bene[1]. Per ogni metrica ci si è basati sulla media delle simulazioni effettuate, tenendo in conto il valore massimo e il valore minimo per esaminare anche l'entità della variazione dei risultati.
La necessità di misurarle entrambe viene dal fatto che essendo le classi potenzialmente sbilanciate, dato che il dataset è non-IID, era importante valutare se questa caratteristica potesse incidere in modo rilevante sulle capacità di previsione del modello.

- **Risorse di calcolo**
  Per quel che riguarda le risorse di calcolo queste erano fornite da Google in cloud, grazie al servizio di Google Colab[15]. Tutte le simulazioni sono state eseguite sulla stessa unità di calcolo: la CPU, i cui numero di core era due. La quantità di memoria utilizzata era sempre pari a 8GB.

4.2 Analisi dei risultati

Prima di analizzare i risultati ottenuti è bene comprendere dalle caratteristiche del sistema ibrido quali possano essere le aspettative sui vari esperimenti effettuati tramite le simulazioni.
Di certo il Single Learning sarebbe risultata la tecnica meno performante di tutte perché ogni client avrebbe contato solo sulle proprie potenzialità. Infatti di per sé questa tecnica di apprendimento è stata inserita in questo studio per fornire una base dalla quale partire, il limite inferiore delle prestazioni che l'addestramento di un modello potesse fornire.
La cosa più interessante da scoprire è come le performance del processo di apprendimento variano in base al variare del grado di ibridazione. Soprattutto scoprire scoprire la differenza che si osserva prendendo in considerazione i due casi estremi di un sistema ibrido del genere, cioè un apprendimento completamente federato e un apprendimento totalmente centralizzato.
L'analisi effettiva dei dati verrà affrontata prendendo in considerazione una metrica per volta e esaminando tutti i risultati che la riguardano delle simulazioni di 3 e 5 round. Verranno presi in considerazione il gap massimo tra le due tecniche che si pongono agli estremi e come cambiano le tendenze di crescita o decrescita della sequenza di valori. Inoltre, se il trend è pressocché lineare è vantaggioso esprimere la relazione che c'è tra grado di ibridazione e risultati della simulazione tramite il metodo della regressione lineare. Esso è un in statistica un metodo di stima del valore atteso che esprime la sequenza dei punti tramite la funzione di una retta del tipo $X = mZ + b$, dove X sarà la variabile dipendente: il livello di ibridazione; Z la variabile indipendente: i valori dei risultati; m è la pendenza o coefficiente angolare della retta; b è l'intercetta[43]. Il valore che per questa analisi risulta più rilevante è il coefficiente angolare che ci può far comprendere la velocità di crescita o decrescita della relazione tra le due variabili.

4.2.1 Tempo di esecuzione

Studiando il tempo di esecuzione della simulazione e si nota con chiarezza nelle Figure 4.1 e 4.2 un trend che era possibile aspettarsi e che verrà comprovato anche da tutti

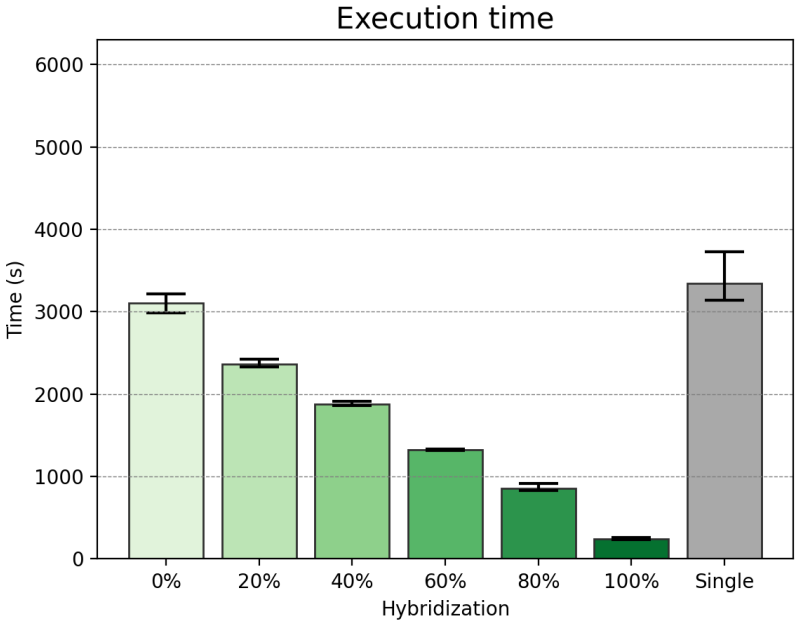

Figura 4.1: Tempo di esecuzione in 3 round

gli altri risultati. La tendenza osservata è che all'aumentare del grado di ibridazione, aumenta in modo potremmo dire lineare anche il tempo di esecuzione della simulazione. Che la simulazione impieghi il tempo massimo per il Single Learning era prevedibile, infatti quel valore serve come standard minimo. È interessante però analizzare il massimo scarto di tempo che si ha tra il FL classico e il ML centralizzato che è pari a 2865 secondi nei 3 round e 4617 secondi nei 5 round, con un decremento percentuale del 92.22% e del 92.41%, rispettivamente. Il notevole miglioramentoè dovuto al fatto che si passa da addestrare 100 client ad addestrarne solamente uno, perciò è un risultato anche comprensibile, sotto un certo punto di vista. Questo perché nel sistema simulato non c'era parallelismo e quindi veniva eseguito un client alla volta.

Approfondiamo ora il trend di miglioramento tra ogni tecnica di apprendimento e quella con il grado di ibridazione maggiore subito successiva, per intenderci tra Federated al 100% e Federated all'80% e così via a coppie. Guardando le figure 4.1 e 4.2 si nota una decrescita lineare costante tra i diversi livelli di ibridazione. Ma se entriamo nel dettaglio è curioso andare a guardare l'andamento del decremento dei valori. Per le simulazioni

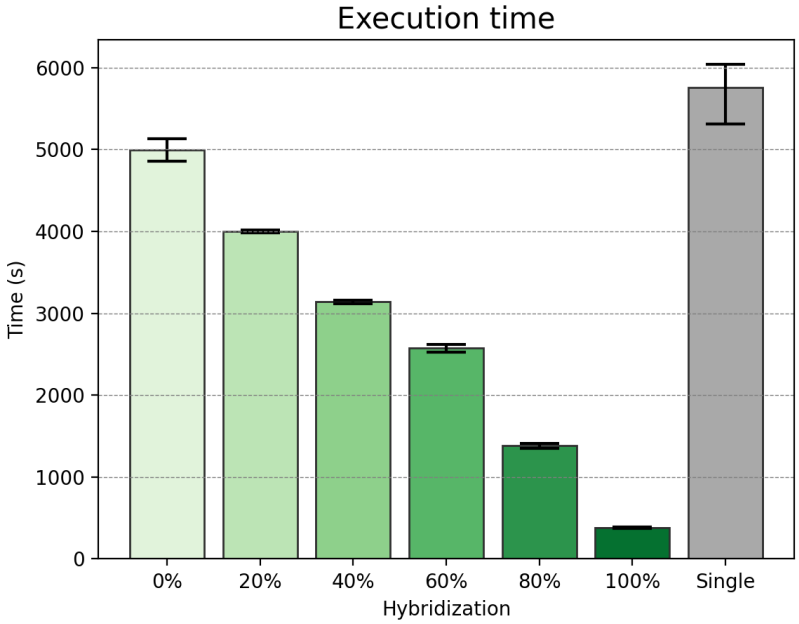

Figura 4.2: Tempo di esecuzione in 5 round

da 3 round la tendenza di decrescita del tempo è quasi lineare tra i primi due per poi iniziare a impennarsi. Infatti si parte da circa un -24%, tra FL e FL con $X = 20$, la tendenza di decrescita diminuisce nel successivo confronto con un -20% per poi ritornare su un trend di forte crescita fino ad arrivare a -72% del Centralized. Più o meno lo stesso andamento si può osservare nei risultati delle simulazioni da 5 round in cui ci si stabilizza intorno al 20% nei primi confronti, il valore minimo è riscontrato tra $X = 40$ e $X = 60$ con decremento del -18% e poi la tendenza schizza fino al 73%. Anche per questi trend sono indirizzabili ad un motivo scatenante cioè che il tempo di esecuzione pian piano diminuisce e le variazioni di valore si fanno più significative.
Visto che la tendenza dei risultati si rivela essere circa lineare uno strumento statistico molto efficace per descrivere l'andamento si rivela la regressione lineare. Dai dati recuperati delle simulazioni con 3 e con 5 round, possiamo ottenere tre coefficienti di regressione lineare, che ci aiutano a descrivere l'evoluzione del tempo di esecuzione in relazione al grado percentuale di ibridazione:

- Simulazione di 3 round: $f(x) = -27.75x + 3015.86$

- Simulazione di 5 round: $f(x) = -44.98x + 4995.25$

- Media delle simulazioni: $f(x) = -36.37x + 4005.56$

Dando un occhiata ai grafici, non si nota questa differenza nella pendenza negativa della decrescita, poiché è una caratteristica osservabile con più facilità dai valori numerici. Però, osservando le barre e anche la scala del tempo, cioè l'asse y del grafico, si nota che il gap tra il massimo e il minimo è più accentuato nella Figura 4.2 rispetto che nella Figura 4.1. Si può constatare dai valori dei coefficienti angolari delle rette che nella simulazione con 5 round la decrescita è più accentuata che in quella da 3 round, perciò, il tempo di esecuzione del sistema di FLI diminuisce di più se si eseguono apprendimenti automatici con un numero di round più alto.
Infine, un ultimo aspetto su cui vale la pena soffermarsi brevemente è la variazione della lunghezza temporale delle simulazioni: più tempo impiega una tecnica di apprendimento ad eseguire tutto il processo, più c'è possibilità che questo sia variabile. Le eccezioni, per tipo di simulazione, sono solo una su sette che sono troppo poche rispetto al resto, il che ci permette di far valere questa affermazione.
Un ultima analisi sul tempo di esecuzione è stata calcolare la media e la deviazione standard dei risultati di tutte le tecniche ibride, per avere un idea della differenza di prestazioni di un qualsiasi sistema ibrido e il classico FL. Per simulazioni da 3 round la media del FL ibrido è pari a 1332 secondi (s), con una deviazione standard di 835 s, in confronto al FL che impiega 3107 s. Per simulazioni con 5 round il FL è impegnato per 4052 s contro la media di 2296 con una dispersione di 1431.
Possiamo concludere affermando che se si considerava un esecuzione concorrente dei client le differenze sarebbero state meno marcate. Quindi questa analisi in particolare riguarda questa specifica simulazione configurata in questo modo e non il sistema generale del FLI in generale. Riguardo a questa simulazione quindi si può dichiarare che all'aumento del livello di ibridazione, il tempo di esecuzione della simulazione diminuiva in modo proporzionale. Il decremento del periodo di esecuzione dal FL alle tecniche ibride dal 20 al 100%, considerando una media dei valori, si posiziona tra il -21% e il -92%.

4.2.2 Accuratezza

Esaminando le prestazioni del modello nella misura dell'accuratezza osserviamo che nei grafici delle Figure 4.3 e 4.4 le barre seguono un andamento non lineare come quello del tempo, ma piuttosto un andamento logistico. Questa tendenza crescente non lineare dei dati si poteva prevedere, dato che è un comportamento proprio delle relazioni tra prestazioni ed esperienza, dove in questo caso le prestazioni sono l'accuratezza del sistema di apprendimento e l'esperienza è rappresentata dal grado di ibridazione del sistema. Tale andamento è rappresentabile tramite una curva di apprendimento, appunto, in una sua forma specifica chiamata aumento esponenziale. L'accuratezza in questo caso può avvicinarsi esponenzialmente ad un limite, in modo simile a quello in cui un condensatore si carica, perciò il valore sale molto velocemente all'inizio per poi stabilizzarsi approcciandosi al valore massimo e crescere sempre più lentamente[41].

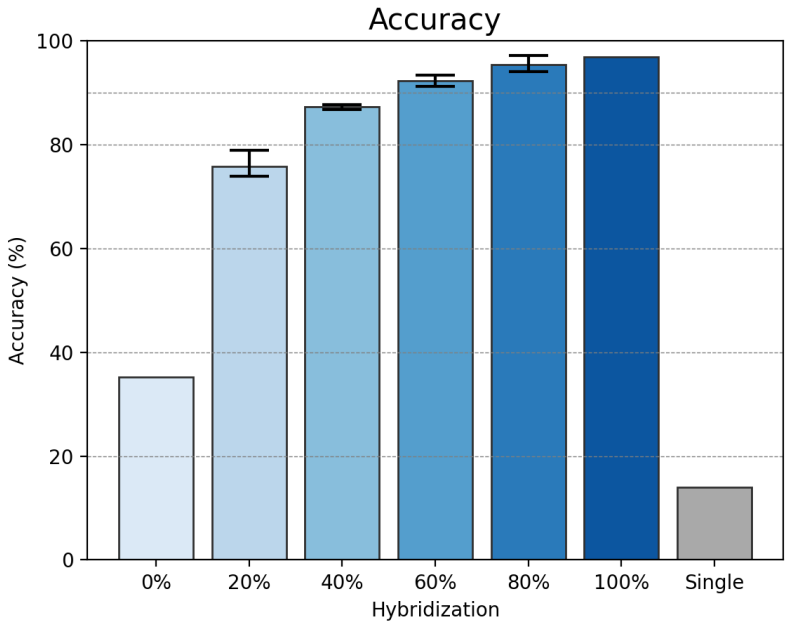

Figura 4.3: Accuracy in 3 round

Tenendo a mente la curva di apprendimento, è estremamente più difficile accrescere le prestazioni di un modello che già ha un alto livello di accuratezza rispetto ad uno

che ha possibilità di incremento maggiori. Infatti le crescite più importanti dell'accuracy tra due tecniche di FL vicine si registrano nel passaggio tra FL e FL ibrido al 20%, in particolare per le simulazioni da 3 round. Per questo caso, infatti, l'aumento di 41 punti percentuali (pp) si rivela in un incremento percentuale di più del doppio (+116%). Mentre con il salire dei round a 5 l'accuracy cresce di un +59%, che comunque rimane valore considerevole. È altrettanto interessante osservare la variazione del gap tra FL e ML totalmente centralizzato. Con 3 iterazioni del sistema il ML riesce ad aumentare l'accuratezza di 62 pp rispetto al FL, con un salto dal 35% al 97%, che si rivela un incremento percentuale del valore dell'accuratezza davvero enorme, pari al +175%. Con 5 round il miglioramento è pari a 47 pp, minore ma comunque altrettanto elevato, con un incremento percentuale che non supera il 100%, ma si assesta sul +90%.
Se vogliamo analizzare in un ottica più generale l'accuracy si può considerare la media dei valori di ogni tecnica a prescindere dai round. In questo caso il divario massimo tra i valori, cioè tra FL e ML, è di 54 pp, che equivale ad un incremento del +124%.
Tirando le somme sull'analisi delle tendenze di crescita è possibile notare che avendo un numero minore di round i valori dell'accuratezza si posizionano in un range maggiore e quindi anche il miglioramento di prestazioni tra una tecnica ed un altra è più accentuato. Se i round aumentano i valori sono tra loro più ravvicinati con un incremento dell'accuracy tra una tecnica ed un altra che è minore ma rimane comunque rilevante.

Si nota come in entrambi i casi il Single Learning performa scarsamente, come era previsto. I client in questione dovevano addestrare in solitario il proprio modello, con oltretutto un numero esiguo di campioni, meno di 100 in media per ognuno. Alcuni potevano anche riuscire delle buone previsioni sul modello, ma chiaramente la condivisione del modello e la sua aggregazione è un aspetto troppo vantaggioso e fondamentale in un sistema federato, che altrimenti farebbe affidamento solo su addestramenti locali su una ridotta quantità di dati .
Riguardo alla variazione, in questo caso come anche per l'F-score, era superfluo considerarla per il Single, il Federated e il Centralized Learning poiché i client erano sempre gli stessi 100, nella stessa sequenza e con gli stessi dataset. Nel caso delle 4 tecniche di apprendimento ibrido i client che adottavano la modalità federata o quella centralizzata erano casuali e quindi sempre diversi. Si osserva come nelle simulazioni da 3 round, in

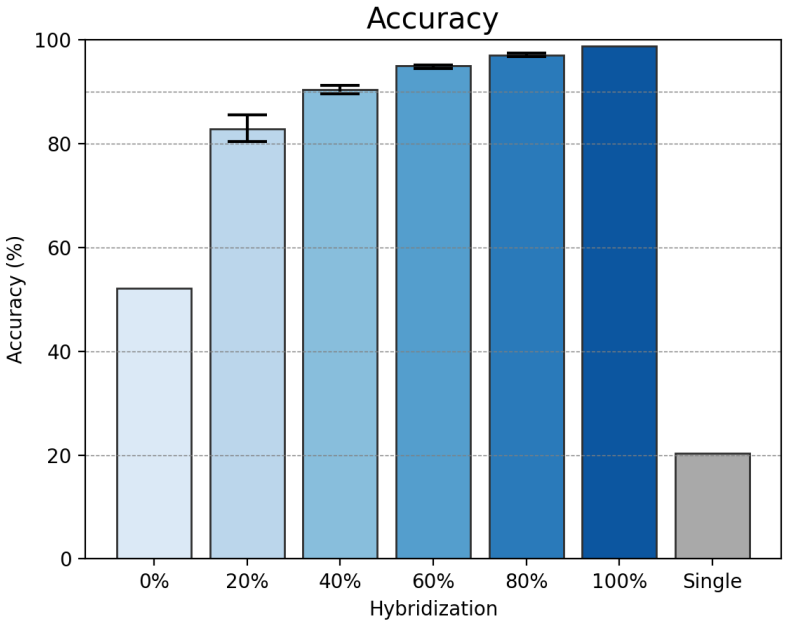

Figura 4.4: Accuracy in 5 round

Figura 4.3, la variazione non segua alcuna regola e sia abbastanza casuale. Mentre invece quando i round salgono a 5 si denota un trend in cui la differenza dei risultati diminuisce in relazione all'aumentare del grado di ibridizzazione, visibile nella Figura 4.4. Questo comportamento è spiegabile dal fatto che meno sono i client selezionati casualmente dal totale per essere accorpati, meno la casualità della scelta incide sul risultato finale dell'accuratezza. Cioè, nell'ibridazione al 20% i client da unire sono 20 su 100 e quindi ci sono molte più possibili combinazioni di tutti gli altri casi. Di conseguenza, constatiamo che la casualità dei client scelti impatta sull'accuratezza del modello. Come per il tempo di esecuzione la variazione segue la stessa tendenza: casuale nei 3 round e decrescente nei 5 round.

Infine un ultima analisi statistica sulle 4 tecniche ibride confrontate con il FL. Essendo la media delle accuracy in 3 e 5 round molto simili utilizziamo la media di tutti i valori. Quindi abbiamo che il FL ottiene un accuratezza media del modello pari al 44% mentre le tecniche ibride raggiungono addirittura il 91% con una deviazione standard di 7 pp. In generale confrontando ogni singolo metodo ibrido implementato con il FL in media si

ottiene un accuracy del modello più alta del 109%, con una variazione del miglioramento che va dal 82% con ibridazione al 20% fino ad un incremento del 124% nel caso del Centralized Learning. Si può quindi affermare che anche un piccolo grado di ibridazione si riflette in un grande miglioramento delle prestazioni del modello.

4.2.3 F-score

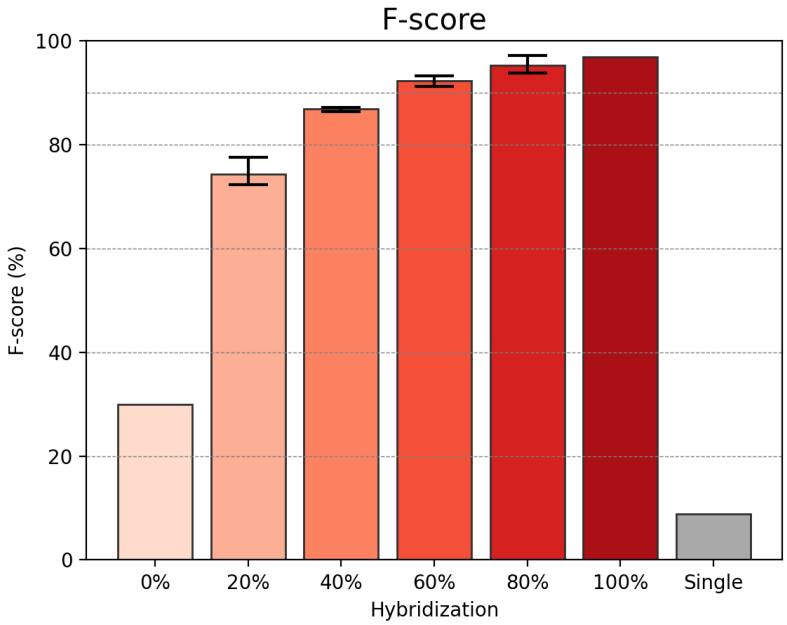

Figura 4.5: F-score in 3 round

Infine analizziamo l'ultima delle tre metriche misurate per il progetto di ibridazione che corrisponde all'F-score, la misura dell'accuratezza di un test. Osservando i grafici relativi a questa metrica nelle Figure 4.5 e 4.6 e confrontandoli con quelli dell'accuracy relativi, rispettivamente in Figura 4.3 e Figura 4.4, notiamo come il loro comportamento sia simile se non uguale in alcuni aspetti. Se consideriamo infatti la tendenza di crescita dei valori notiamo come seguano l'andamento logistico, proprio anche dell'accuratezza. Di conseguenza, possiamo immaginare che tali risultati possano essere rappresentati sotto forma di curva di apprendimento con un ascesa esponenziale con successiva stabilizzazione verso il limite superiore.

Pertanto date le troppo simili caratteristiche tra accuracy e F-score, sarebbe ridondante effettuare un analisi degli stessi aspetti. Ci concentreremo quindi in quali sono le differenze e le relazioni tra le due. Si nota che quando l'accuratezza del modello è bassa, l'F-measure è ancor più inferiore. Il divario maggiore si ha tra il Single Learning e il FL con l'incremento percentuale dell'f-score che nelle simulazioni di 3 round è di quasi due volte e mezzo (+237%), mentre in quelle da 5 round per poco non arriva al doppio, con un +192%. Questo maggior divario tra accuracy e F1 score nelle tecniche di apprendimento meno efficienti si riscontra anche nell'enorme salto che si ha tra FL e ML centralizzato, la cui media registra una crescita di 60 pp pari ad un aumento percentuale del +156%, mentre i singoli valori sono +223% in 3 round e +113% in 5 round.
Facendo un confronto diretto tra i risultati di accuratezza e f-score abbiamo che nel FL classico la differenza è massima tra i due valori e in media è di 5 pp. Man mano che migliorano le prestazioni del modello la discrepanza va diminuendo sempre più fino ad annullarsi, ottenendo valori identici delle due metriche.

A prescindere comunque, la differenza riscontrata è davvero irrisoria, quasi nulla, e in

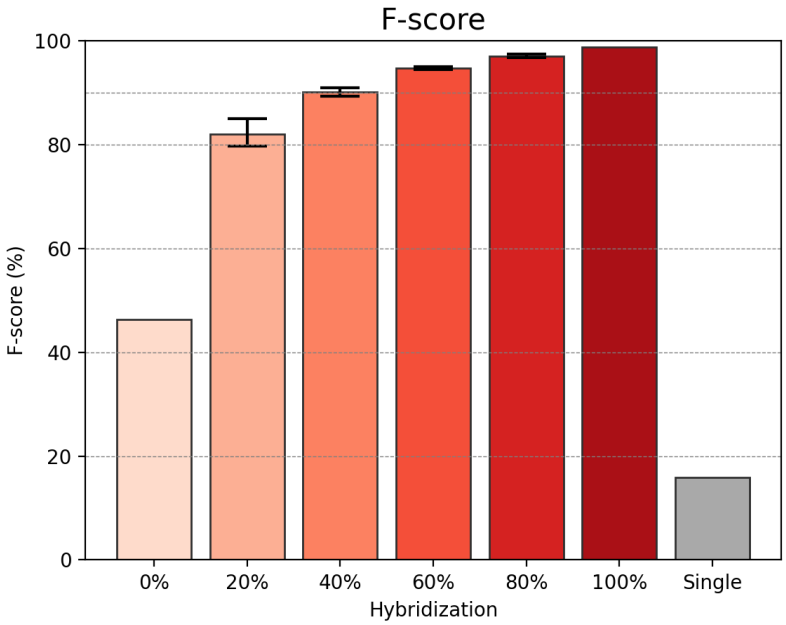

Figura 4.6: F-score in 5 round

generale sono sempre molto vicini i risultati di accuracy e f-score. Questo è dato dal fatto che nonostante i dati di apprendimento potessero essere sbilanciati, data la non-IIDness del dataset FEMNIST utilizzato, alla fine la distorsione non ha avuto influenze sul modello e sulle sue prestazioni. Infatti, molto probabilmente, lo squilibrio dei dati utilizzati per l'addestramento se era presente si è rivelato essere di un'entità davvero irrilevante.
Un ultimo confronto è quello tra il FL classico e tutte le tecniche con più o meno livello di ibridazione. La media di tutti i valori dell'f-score di tali metodi è pari al 91% con una dispersione dei valori di 8 pp a fronte di un f-score del FL che ammonta al 38%. E infatti se si mette in relazione la tecnica completamente federata con ognuna di quelle ibride si riscontra un aumento dell'accuratezza del test in ogni caso pari al doppio del valore del FL. L'incremento va da +105% a +156%, con una media del 138%.

4.3 Miglioramenti

È importante riconoscere i limiti di questa ricerca sul FLI e di conseguenza pensare a delle possibili migliorie per questo sistema. Vediamo quindi i miglioramenti che si potrebbero applicare alla ricerca appena effettuata e a tutta l'infrastruttura che compone l'ambiente di ricerca:

- **Aumentare le risorse di calcolo**
  Possiamo dire che questo sia il punto di partenza che potrebbe permettere un evoluzione del sistema, sotto il punto di vista delle sue capacità e opportunità, affidabilità e diversificazione dei risultati ottenuti, attinenza dell'ambiente simulato con un reale sistema federato.
  Incrementare le risorse computazionali potrebbe permettere al sistema ibrido di gestire più client e/o più dati, velocizzare il tedioso tempo di attesa che si è costretti ad aspettare per raccogliere i risultati al termine della simulazione, parallelizzare l'esecuzione di più client contemporaneamente.

- **Incrementare i client**
  Aumentando i client si potrebbe creare un ambiente sempre più simile ad un sistema federato reale in cui i client possono essere migliaia e arrivare addirittura al milione

di dispositivi. Di conseguenza si potrebbe capire ancora meglio la vera portata del vantaggio di un sistema ibrido applicato ad un contesto reale. Utilizzando Flower questo non sarebbe nemmeno un problema poiché questo framework può arrivare a gestire 1000 client concorrenti su un totale di 1 milione[3].
In questo modo però si dovranno anche eseguire un numero maggiore di simulazione per ogni tipo, perché la scelta casuale dei client centralizzati avrebbe molte più combinazioni.

- **Accrescere la dimensione dei dati**
  Un alternativa all'incremento dei client potrebbe essere l'aumento dei loro dataset locali. In questo modo sarebbe possibile analizzare come si comporta il sistema quando la mole di dati gestita è molto maggiore. Supponendo di utilizzare un dataset non-IID, sappiamo che aumentando la quantità di dati aumenta anche la possibilità di client anomali e distorsione dei dati e delle classi[45]. Di conseguenza, si potrebbe valutare come cambiano i risultati quando la distorsione dei dati è maggiore e quindi capire meglio come si comporta questo sistema ibrido con i dati non-IID

- **Variare il dataset**
  Si potrebbe testare questo sistema con dataset differenti per poter confrontare i diversi risultati che si ottengono dal sistema in base al dataset utilizzato, mantenendo tutte le restanti caratteristiche e configurazioni fisse.
  O ancora si potrebbe valutare il comportamento del sistema quando i dati sono IID e di conseguenza confrontare le differenze con i risultati ottenuti dallo stesso dataset ma nella forma non-IID.

- **Estendere i livelli di ibridazione**
  Si potrebbero inserire più intervalli nel grado di ibridazione del sistema così da avere risultati più consistenti e di conseguenza grafici più esplicativi e completi, con cui comprendere ancora meglio le tendenze del sistema. Oltretutto si potrebbe esplorare con più precisione a quale grado di ibridazione il sistema FLI diventa con chiarezza più efficiente rispetto al FL classico.

- **Variare il numero di round**
  Facendo simulazioni con un numero di round molto più variabile si potrebbe analizzare come si comporta il FLI con un alto numero di round e se rimane ancora conveniente sotto il punto di vista dell'efficienza.

Queste appena affrontate sono tutte le possibili modifiche e miglioramenti che si potrebbero applicare a questo sistema nell'ambito della ricerca, sempre utilizzando lo strumento delle simulazioni fornito da Flower.
Flower però è progettato appositamente per fare ricerche sperimentali tramite simulazioni oppure ricerche di sistema su una vasta rete di dispositivi reali. Oltretutto la migrazione da un ambiente ad un altro è facilitato dal framework stesso[3]. Poter testare l'ambiente di FLI in una reale rete federata di edge device costituirebbe una sperimentazione fondamentale e conclusiva di questo sistema. Sia perché applicato a contesti reali potrebbe creare dei vantaggi notevoli nell'ambito dei sistemi federati, sia perché se ne potrebbero comprendere a pieno le capacità, l'utilità e le possibili limitazioni e punti deboli. Solamente applicandolo ad uno scenario concreto si potrebbe davvero testare un sistema, sul quale fino a quel momento sono stati effettuati solo studi sperimentali.

Conclusioni

Questo elaborato aveva come obiettivo l'implementazione e lo studio di un sistema di apprendimento che ibridasse FL e ML centralizzato in modo tale da far convergere i vantaggi di entrambe le tecniche in un unico ambiente.
La ricerca prevedeva la ripetizione di esecuzione e conseguente analisi dettagliata di simulazioni del Federated Learning Ibrido fornendo gli stessi dataset e identiche configurazioni, ad eccezione del numero di round e del tasso di ibridazione del sistema. I round potevano essere 3 o 5 mentre le tecniche di apprendimento erano FL, ML classico o metodi basato sul FL che incorporavano una parte del sistema che operava in modo centralizzato. In questo ultimo caso il FL era ibrido al 20%, 40%, 60% o 80%.
Lo studio ha prodotto degli esiti abbastanza solidi per poter arrivare a conclusioni valide e consistenti. Possiamo affermare che ibridare il FL con un unità centrale porta enormi vantaggi all'efficienza del sistema e in parte anche alla velocità. Considerando che i client che decidono di condividere i dati grezzi lo fanno con il loro libero arbitrio, questo sistema porta soltanto benefici all'addestramento del modello e, di riflesso, anche ai dispositivi che vi partecipano, i quali giovano delle eccellenti performance. Oltretutto si evidenzia come anche un ridotto grado di ibridazione del sistema possa aumentare di molto l'accuratezza del modello addestrato
Queste asserzioni sono pienamente comprovate dai dati che ne evidenziano ancora di più il notevole beneficio di un'ibridazione del FL. Infatti, rispetto al FL se si esegue l'apprendimento con un metodo ibrido si può ottenere una riduzione del tempo di esecuzione dell simulazione del 55%. Ancora più notevole è il risultato raggiunto dall'accuratezza del modello, considerando accuracy e F-score insieme, che in media incrementa del 124%.
In generale, considerando la totalità degli aspetti relativi che influiscono sul modello, le

tecnologie ibride possono imprimere un miglioramento che va dal 57% fino al 116%, con una media che si assesta sull'89%.

Bibliografia

[1] Stephen Allwright. *F1 score vs accuracy, which is the best metric?* en. Lug. 2022. URL: https://stephenallwright.com/f1-score-vs-accuracy/ (visitato il 28/11/2023).

[2] The Flower Authors. *Flower: A Friendly Federated Learning Framework*. en-GB. URL: https://flower.dev/ (visitato il 26/11/2023).

[3] Daniel J. Beutel et al. *Flower: A Friendly Federated Learning Research Framework*. arXiv:2007.14390 [cs.LG]. Mar. 2022. URL: http://arxiv.org/abs/2007.14390 (visitato il 26/11/2023).

[4] Sebastian Caldas et al. *LEAF: A Benchmark for Federated Settings*. arXiv:1812.01097 [cs.LG]. Dic. 2019. DOI: 10.48550/arXiv.1812.01097. URL: http://arxiv.org/abs/1812.01097 (visitato il 23/11/2023).

[5] Tarin Clanuwat et al. *Deep Learning for Classical Japanese Literature*. arXiv:1812.01718 [cs.LG]. Dic. 2018. DOI: 10.20676/00000341. URL: http://arxiv.org/abs/1812.01718 (visitato il 24/11/2023).

[6] Gregory Cohen et al. *EMNIST: an extension of MNIST to handwritten letters*. arXiv:1702.05373 [cs.CV]. Mar. 2017. DOI: 10.48550/arXiv.1702.05373. URL: http://arxiv.org/abs/1702.05373 (visitato il 02/12/2023).

[7] N. B. T. Digital. *Benefits of Federated Learning Explained*. en-GB. Gen. 2023. URL: https://octaipipe.ai/benefits-of-federated-learning-explained/ (visitato il 23/11/2023).

[8] Python Software Foundation. *Our Documentation — Python.org*. en. URL: https://www.python.org/doc/ (visitato il 26/11/2023).

[9] Jim Frost. *Independent and Identically Distributed Data (IID)*. en-US. Ago. 2020. URL: https://statisticsbyjim.com/basics/independent-identically-distributed-data/ (visitato il 23/11/2023).

[10] GeeksforGeeks - abhishekaslk. *Epoch in Machine Learning*. en. Section: Data Science. Gen. 2023. URL: https://www.geeksforgeeks.org/epoch-in-machine-learning/ (visitato il 29/11/2023).

[11] Vito Gentile. *Google Colab per il Machine Learning: cos'è e come si usa*. it. Dic. 2019. URL: https://www.html.it/articoli/google-colab-per-il-machine-learning-cose-e-come-si-usa/ (visitato il 26/11/2023).

[12] Everton Gomede. *Independent and Identically Distributed (IID) in Machine Learning: Assumptions and Implications — by Everton Gomede, PhD — Medium*. Feb. 2023. URL: https://medium.com/@evertongomede/independent-and-identically-distributed-iid-in-machine-learning-assumptions-and-implications-930ee9821e14 (visitato il 23/11/2023).

[13] Google. *Classificazione: accuratezza — Machine Learning*. it. URL: https://developers.google.com/machine-learning/crash-course/classification/accuracy?hl=it (visitato il 28/11/2023).

[14] Google. *Google Colab -FAQ*. URL: https://research.google.com/colaboratory/faq.html (visitato il 26/11/2023).

[15] Google. *Google Colaboratory*. it. URL: https://colab.research.google.com/ (visitato il 26/11/2023).

[16] Google Brain. *Federated Learning for Image Classification — TensorFlow Federated*. en. URL: https://www.tensorflow.org/federated/tutorials/federated_learning_for_image_classification (visitato il 01/12/2023).

[17] GitHub Inc. *GitHub*. en. URL: https://github.com (visitato il 24/11/2023).

[18] Team I. A. Italia. *Come Creare Un Dataset per il tuo Progetto Di Machine Learning (ML) o Deep Learning (DL) — Intelligenza Artificiale Italia Blog*. it. Giu. 2021. URL: https://www.intelligenzaartificialeitalia.net/post/come-creare-

un-dataset-per-il-tuo-progetto-di-machine-learning-ml-o-deep-learning-dl (visitato il 23/11/2023).

[19] Dr Amita Kapoor. *The Power of IID Data in Machine Learning and Deep Learning Models*. en. Mar. 2023. URL: https://medium.com/@kapooramita/the-power-of-iid-data-in-machine-learning-and-deep-learning-models-49651afb7882 (visitato il 23/11/2023).

[20] *LEAF 0.1 documentation*. URL: https://leaf.cmu.edu/build/html/install/get_leaf.html (visitato il 24/11/2023).

[21] *LEAF: A Benchmark for Federated Settings*. original-date: 2018-10-26. Nov. 2023. URL: https://github.com/TalwalkarLab/leaf (visitato il 24/11/2023).

[22] Yann LeCun, Corina Cortes e Chris Burges. *MNIST handwritten digit database*. URL: http://yann.lecun.com/exdb/mnist/ (visitato il 23/11/2023).

[23] Qun Li et al. *Vertical Federated Learning: Taxonomies, Threats, and Prospects*. arXiv:2302.01550 [cs.LG]. Feb. 2023. DOI: 10.48550/arXiv.2302.01550. URL: http://arxiv.org/abs/2302.01550 (visitato il 23/11/2023).

[24] Linux Foundation. *PyTorch*. en. URL: https://pytorch.org (visitato il 26/11/2023).

[25] Filomena Mauriello. «Tecniche di ricampionamento per dataset con classi di risposta sbilanciate. Una proposta metodologica per dataset con predittori di natura numerica e categorica». it. Tesi di dott. Corso Umberto I, 40 - 80138 Napoli: Università degli Studi di Napoli Federico II, mar. 2014. URL: http://www.fedoa.unina.it/9890/2/Filomena%20Mauriello%20Tesi%20di%20dottorato.pdf.

[26] Pooja N. *The Evolution of Federated Learning — Tibil Solutions*. en-US. Ott. 2023. URL: https://tibilsolutions.com/evolution-of-federated-learning/ (visitato il 23/11/2023).

[27] Nazar Kvartalny. *4 Reasons Why is Python Used for Machine Learning — Inoxoft*. en-US. Lug. 2022. URL: https://inoxoft.com/blog/why-use-python-for-machine-learning/ (visitato il 26/11/2023).

[28] Travis Oliphant. *NumPy*. en. URL: https://numpy.org/ (visitato il 26/11/2023).

[29] Andrea Provino. *Horizontal vs Vertical vs Transfer Federated Learning*. it. Ott. 2020. URL: https://www.andreaprovino.it/horizontal-vs-vertical-vs-transfer-federated-learning (visitato il 23/11/2023).

[30] pulplearning. *Federated Learning: cos'è e come funziona*. it. Lug. 2023. URL: https://pulplearning.altervista.org/federated-learning-cose-e-come-funziona/ (visitato il 23/11/2023).

[31] G. Pradeep Reddy e Y. V. Pavan Kumar. «A Beginner's Guide to Federated Learning». In: *2023 Intelligent Methods, Systems, and Applications (IMSA)*. Lug. 2023, pp. 557–562. DOI: 10.1109/IMSA58542.2023.10217383. URL: https://ieeexplore.ieee.org/abstract/document/10217383 (visitato il 23/11/2023).

[32] Alessandro Rizzuto. *Deep learning e dataset sintetici: come affrontare la mancanza di dati*. Section: Intelligenza Artificiale. Nov. 2020. URL: https://www.bigdata4innovation.it/intelligenza-artificiale/deep-learning-e-dataset-sintetici-come-affrontare-la-mancanza-di-dati/ (visitato il 23/11/2023).

[33] Yesha Shastri. *A Step-by-Step Guide to Federated Learning in Computer Vision*. en. Feb. 2023. URL: https://www.v7labs.com/blog/federated-learning-guide (visitato il 23/11/2023).

[34] Eli Stevens, Luca Antiga e Thomas Viehmann. *Deep Learning with PyTorch*. en. Google-Books-ID: fff1DwAAQBAJ. Simon e Schuster, ago. 2020. ISBN: 978-1-61729-526-3.

[35] tao-shen. *FEMNIST_pytorch*. original-date: 2020-04-07T12:10:20Z. Set. 2023. URL: https://github.com/tao-shen/FEMNIST_pytorch (visitato il 24/11/2023).

[36] Taylor Karl. *6 Reasons Why Is Python Used for Machine Learning*. en-US. Ago. 2023. URL: https://unitedtraining.com/resources/blog/why-is-python-used-for-machine-learning (visitato il 26/11/2023).

[37] Kang Wei et al. *Vertical Federated Learning: Challenges, Methodologies and Experiments*. arXiv:2202.04309 [cs.LG]. Feb. 2022. DOI: 10.48550/arXiv.2202.04309. URL: http://arxiv.org/abs/2202.04309 (visitato il 23/11/2023).

[38] Wikipedia. *CIFAR-10*. en. Page Version ID: 1166845013. Lug. 2023. URL: https://en.wikipedia.org/wiki/CIFAR-10 (visitato il 23/11/2023).

[39] Wikipedia. *Conda (package manager)*. en. Page Version ID: 1183794577. Nov. 2023. URL: https://en.wikipedia.org/w/index.php?title=Conda_(package_manager)&oldid=1183794577 (visitato il 26/11/2023).

[40] Wikipedia. *F-score*. en. Page Version ID: 1174585267. Set. 2023. URL: https://en.wikipedia.org/w/index.php?title=F-score&oldid=1174585267 (visitato il 28/11/2023).

[41] Wikipedia. *Learning curve*. en. Page Version ID: 1177429602. Set. 2023. URL: https://en.wikipedia.org/w/index.php?title=Learning_curve&oldid=1177429602 (visitato il 29/11/2023).

[42] Wikipedia. *MNIST database*. en. Page Version ID: 1174503124. Set. 2023. URL: https://en.wikipedia.org/w/index.php?title=MNIST_database (visitato il 23/11/2023).

[43] Wikipedia. *Regressione lineare*. it. Page Version ID: 134870822. Ago. 2023. URL: https://it.wikipedia.org/w/index.php?title=Regressione_lineare&oldid=134870822#Regressione_lineare_semplice (visitato il 29/11/2023).

[44] Wikipedia. *Visual Studio Code*. it. Page Version ID: 135503778. Set. 2023. URL: https://it.wikipedia.org/w/index.php?title=Visual_Studio_Code&oldid=135503778 (visitato il 26/11/2023).

[45] C. Xiao e S. Wang. «An Experimental Study of Class Imbalance in Federated Learning». en. In: *2021 IEEE Symposium Series on Computational Intelligence (SSCI)*. arXiv:2109.04094 [cs.LG]. Dic. 2021, pp. 1–7. DOI: 10.1109/SSCI50451.2021.9660072. URL: http://arxiv.org/abs/2109.04094 (visitato il 24/11/2023).

[46] Han Xiao, Kashif Rasul e Roland Vollgraf. *Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms*. arXiv:1708.07747 [cs.LG]. Set. 2017. DOI: 10.48550/arXiv.1708.07747. URL: http://arxiv.org/abs/1708.07747 (visitato il 24/11/2023).

[47] Qiang Yang et al. «Horizontal Federated Learning». In: *Federated Learning*. Cham: Springer International Publishing, 2020, pp. 49–67. ISBN: 978-3-031-01585-4. DOI: 10.1007/978-3-031-01585-4_4. URL: https://doi.org/10.1007/978-3-031-01585-4_4.

[48] Xue Ying. «An Overview of Overfitting and its Solutions». en. In: *Journal of Physics: Conference Series* 1168 (feb. 2019), p. 1. ISSN: 1742-6588, 1742-6596. DOI: 10.1088/1742-6596/1168/2/022022. URL: https://iopscience.iop.org/article/10.1088/1742-6596/1168/2/022022 (visitato il 25/11/2023).

[49] Xinwei Zhang et al. *Hybrid Federated Learning: Algorithms and Implementation*. arXiv:2012.12420 [cs.LG]. Feb. 2021. DOI: 10.48550/arXiv.2012.12420. URL: http://arxiv.org/abs/2012.12420 (visitato il 26/11/2023).

[50] Yue Zhao et al. «Federated Learning with Non-IID Data». In: (2018). arXiv:1806.00582 [cs.LG]. DOI: 10.48550/arXiv.1806.00582. URL: http://arxiv.org/abs/1806.00582 (visitato il 23/11/2023).

[51] Hangyu Zhu, Haoyu Zhang e Yaochu Jin. «From federated learning to federated neural architecture search: a survey». en. In: *Complex & Intelligent Systems* 7.2 (apr. 2021), pp. 639–657. ISSN: 2198-6053. DOI: 10.1007/s40747-020-00247-z. URL: https://doi.org/10.1007/s40747-020-00247-z (visitato il 26/11/2023).

Ringraziamenti

Ringrazio fortemente tutta la mia famiglia, i miei genitori in particolare, che mi hanno sempre spronato e sostenuto, nonostante tutto. Grazie soprattutto per avermi donato questa opportunità, ve ne sono riconoscente.
Un ringraziamento anche a tutte le persone che hanno condiviso con me questo percorso.