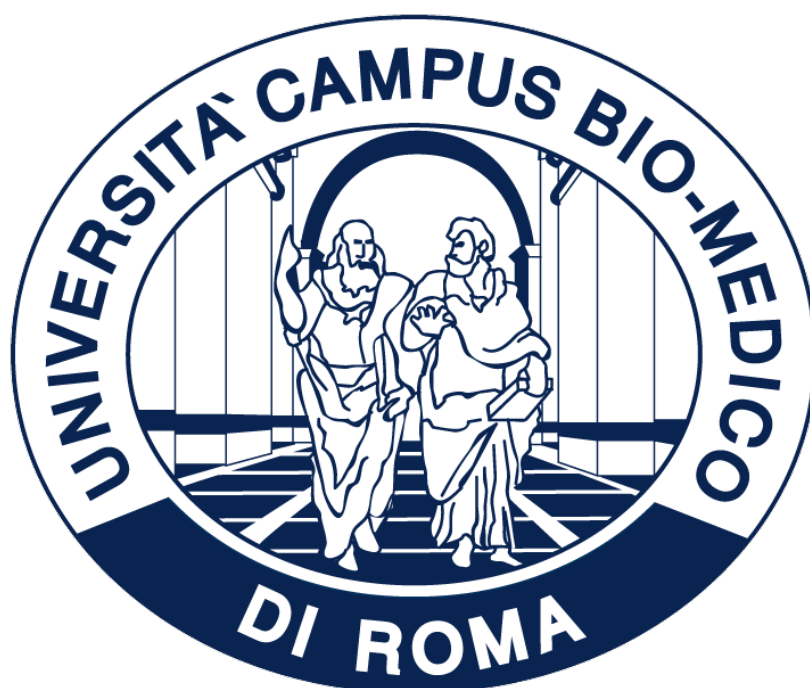


# Un'architettura Web3 per il monitoraggio e auditing dell'apprendimento AI Federato



**Relatore:**

Prof. Mario Merone

**Candidato:**

Riccardo Casaula

**Correlatore:**

Ing. Lorenzo Petrosino

Anno Accademico 2024–2025

# 1 Introduzione

L'intelligenza artificiale (AI) sta rivoluzionando settori chiave come la medicina, la finanza e l'industria manifatturiera, migliorando processi decisionali, diagnostici e operativi. In ambito medico, l'AI consente diagnosi più tempestive e accurate, personalizzazione dei trattamenti e gestione ottimizzata delle risorse ospedaliere. Nel contesto economico, essa facilita l'automazione delle analisi finanziarie, il rilevamento delle frodi e la previsione dei trend di mercato. Tuttavia, la crescente adozione di queste tecnologie solleva questioni critiche in termini di sicurezza e privacy dei dati sensibili. Uno degli aspetti più problematici dell'AI riguarda la tutela della privacy, specialmente quando i modelli vengono addestrati su dati personali. La crescente attenzione verso la protezione dei dati ha portato all'introduzione di normative specifiche, tra cui l'AI Act europeo [1]. Questa regolamentazione enfatizza la necessità di garantire trasparenza e verificabilità nei processi di apprendimento, imponendo che i modelli AI rispettino rigorosi criteri di sicurezza e conformità normativa.

## 1.1 Federated Learning: Un Approccio Privacy-Preserving

Il Federated Learning (FL) si propone come una soluzione innovativa per la protezione della privacy nell'addestramento di modelli AI [2]. Questo paradigma consente di sviluppare modelli direttamente sui dispositivi degli utenti (peer), evitando il trasferimento di dati a server centralizzati. Grazie a questa architettura, le informazioni rimangono distribuite, riducendo il rischio di esposizione o compromissione dei dati. Un ruolo cruciale in questa architettura è svolto dall'aggregatore, un'entità responsabile della raccolta e dell'elaborazione degli aggiornamenti dei modelli locali senza accedere direttamente ai dati sensibili. L'aggregatore calcola un modello globale aggiornato sulla base dei contributi ricevuti dai dispositivi distribuiti, garantendo efficienza nell'addestramento e coerenza nei risultati, pur mantenendo la decentralizzazione e la sicurezza del processo. Tuttavia, il Federated Learning pone ancora sfide significative in termini di verificabilità e trasparenza dei processi di apprendimento. e tecnologie a registri distribuiti (DLT), e in particolare le blockchain, offrono una soluzione efficace per garantire sicurezza, integrità e trasparenza nella gestione dei dati. La prima blockchain decentralizzata, Bitcoin [3], ha dimostrato come sia possibile assicurare l'immutabilità e la sicurezza delle transazioni senza il bisogno di un'autorità centrale. Applicando questi stessi principi all'intelligenza artificiale, le blockchain possono contribuire a tracciare e validare il processo di apprendimento dei modelli, garantendo maggiore trasparenza, affidabilità in conformità alle normative.

## 1.2 Smart Contract e Applicazioni Web3 per l'Auditing AI

Web3 [4] è un insieme di protocolli e strumenti che consentono di interagire con la blockchain, senza la necessità di intermediari. Tramite tecnologie come gli smart contract e i protocolli peer-to-peer, Web3 permette agli utenti di accedere e gestire dati e transazioni in maniera autonoma e verificabile. Questo approccio favorisce la creazione di applicazioni decentralizzate (DApp), offrendo un ambiente più sicuro, trasparente e resistente alla censura rispetto ai modelli centralizzati. Web3 fornisce un nuovo paradigma di interazione con i sistemi distribuiti, abilitando una gestione diretta delle risorse digitali sulla blockchain.

All'interno di questa infrastruttura, gli *smart contract* [5] giocano un ruolo cruciale. Questi si comportano come veri e propri programmi "auto-eseguibili", operando su blockchain, permettono l'automazione e la verifica di operazioni complesse, garantendo trasparenza e immutabilità. Tuttavia, il loro utilizzo non esclude la possibilità di supervisione esterna: al contrario, grazie alla registrazione inalterabile delle operazioni sulla blockchain, essi facilitano l'auditing da parte di enti esterni. Questo li rende strumenti particolarmente adatti per certificare e monitorare il processo di apprendimento AI federato, assicurando che ogni fase sia verificabile e conforme agli standard di sicurezza e regolamentazione.

Le applicazioni decentralizzate (*DApp*) [6] sono costruite sulla base degli *smart contract*, che ne costituiscono la logica operativa e regolano il loro funzionamento. In pratica, gli *smart contract* fungono da scheletro delle *DApp*, automatizzando le regole e le interazioni al loro interno. Grazie a questa architettura, le operazioni di apprendimento AI federato possono essere registrate e verificate in modo immutabile, garantendo trasparenza e conformità normativa. L'integrazione di Web3 con il Federated Learning permette infatti di creare un ecosistema decentralizzato, sicuro e verificabile, in cui la tracciabilità e la trasparenza offerte dagli *smart contract* rispondono alle esigenze di regolamentazione imposte dall'AI Act e da altre normative emergenti.

## 2 Obiettivi

L'obiettivo di questa tesi è sviluppare un'architettura Web3 per l'auditing e il monitoraggio dell'apprendimento AI federato [7]. La soluzione proposta integra il Federated Learning con la tecnologia blockchain per garantire la trasparenza e la tracciabilità del processo di addestramento. Attraverso l'uso di smart contract, ogni fase dell'apprendimento viene registrata in modo immutabile, permettendo la verifica da parte di enti esterni a costo zero. Il processo di auditing è semplice e non richiede competenze tecniche informatiche, rendendo l'accesso alla verifica più inclusivo ed efficace. Questo approccio fornisce un'infrastruttura sicura e decentralizzata, in conformità con le normative vigenti.

## 3 Descrizione lavoro svolto

### 3.1 Modello ad apprendimento Federato

Per sviluppare l'architettura proposta, si è scelto un task di *training* noto come benchmark in letteratura, ovvero la classificazione del dataset MNIST [8]. MNIST (Modified National Institute of Standards and Technology) è un dataset ampiamente utilizzato nel *machine learning* per il riconoscimento delle cifre scritte a mano. Contiene 60.000 immagini di addestramento e 10.000 immagini di test, ognuna delle quali rappresenta una cifra numerica (0-9) in formato 28x28 pixel in scala di grigi.

Per implementare il *Federated Learning*, è stato creato un ambiente di test costituito da 5 *peer* (o *client*) e un aggregatore, ognuno dei quali esegue localmente il *training* sui propri dati. I *peer* (o *client*) sono i nodi che detengono i dati a livello locale e addestrano autonomamente il modello su tali dati. Al termine di ogni fase di addestramento, ciascun *peer* condivide unicamente i parametri appresi (e non i dati originali) con l'aggregatore, che a sua volta combina questi parametri per aggiornare il modello globale. Un esempio pratico in ambito medico può aiutare a comprendere meglio questi ruoli: immaginando

un insieme di ospedali che desiderano addestrare un modello di diagnosi medica senza condividere direttamente i dati dei pazienti (ad esempio, per ragioni di privacy), ognuno di essi agirebbe come un *peer*. Ciascun ospedale addestrerebbe il proprio modello sulle cartelle cliniche locali e invierebbe solo i parametri appresi (e non i dati sensibili) all'aggregatore, il quale, mettendo insieme i parametri dei vari ospedali, costruirebbe un modello diagnostico globale più accurato, senza che le informazioni sensibili dei pazienti escano mai dai singoli ospedali.

Parallelamente, è stato sviluppato un algoritmo in Python per gestire la connessione alla rete tramite Web3 come mostrato in figura 1, consentendo un'integrazione efficiente con la *blockchain* ai fini della registrazione e verifica dei dati di *training*. Inoltre, sono state progettate strutture dati ottimizzate per raccogliere e organizzare le informazioni critiche, assicurando che ogni aggiornamento fosse *auditabile* e facilmente verificabile. Questo approccio garantisce un registro trasparente delle metriche di addestramento, fornendo una base solida per il monitoraggio e l'analisi dell'intero processo di *Federated Learning*.

```

338
339     peer_infos = [
340         peer_info_1,
341         peer_info_2,
342         peer_info_3,
343         peer_info_4,
344         peer_info_5
345     ]
346     for (client_info, peer_info) in zip(client_contracts, peer_infos):
347         account = client_info["account"]
348         contract = client_info["contract"]
349
350         # L'unico parametro che cambia è la stringa peer_info
351         # Il round rimane fisso a ROUND_NUMBER
352         tx_hash = contract.functions.mint(i+1, peer_info).transact({'from': account.address})
353         print("Tx inviata per il peer con info:", peer_info)
354         print("Tx hash:", tx_hash.hex())
355         receipt = w3.eth.wait_for_transaction_receipt(tx_hash)
356         print("Tx inclusa nel blocco:", receipt.blockNumber)
357         print("-" * 70)
358
359
360     nonce = w3.eth.get_transaction_count(Aggregator_account.address)
361     tx_hash = A_contract_ABI.functions.mint(hash_avg_weights, roundInfo_str).transact({'from': Aggregator_account.address})
362     print("Tx inviata per l'aggregatore", hash:", tx_hash.hex())
363     receipt = w3.eth.wait_for_transaction_receipt(tx_hash)
364     print("Tx inclusa nel blocco:", receipt.blockNumber)
365     print("Status:", receipt.status)
366

```

Figure 1: Integrazione Web3

### 3.2 Funzione degli smart contract

Successivamente, sono stati sviluppati due smart contract utilizzando Remix [9], un ambiente di sviluppo integrato (IDE) basato sul web, progettato per scrivere, testare e distribuire smart contract sulla blockchain di Ethereum. I due contratti, denominati FedAggregator (2) e FedPeer (3), regolano rispettivamente le operazioni dell'aggregatore e dei client (peer), gestendo la registrazione delle informazioni di training e dei parametri condivisi. Un aspetto estremamente importante dell'implementazione è l'integrazione della distribuzione di OpenZeppelin [10], una libreria fondamentale per lo sviluppo sicuro di smart contract. Grazie ai moduli forniti da OpenZeppelin, è stato possibile implementare gli standard ERC721 [11], utilizzati per la gestione degli NFT e garantire funzionalità critiche come il controllo degli accessi, la sicurezza nelle transazioni e la gestione dei permessi. Lo standard ERC721 è un protocollo per la creazione e gestione di token non fungibili (NFT) sulla blockchain di Ethereum [12]. A differenza dei token fungibili ERC20, che sono intercambiabili tra loro, gli NFT ERC721 sono unici e non divisibili, ciascuno caratterizzato da un identificatore univoco e metadati associati. In questo progetto, ogni

NFT generato rappresenta un round di addestramento del modello AI e contiene informazioni specifiche, come gli aggiornamenti del modello, le metriche di training e il numero del round. Questo garantisce tracciabilità, immutabilità e auditabilità dei dati registrati sulla blockchain. L'adozione dello standard ERC721, combinata con l'uso di OpenZeppelin, ha permesso di implementare un sistema sicuro e conforme alle best practice del settore. L'impiego di contratti pre-auditati ha ridotto il rischio di vulnerabilità, assicurando robustezza e affidabilità all'intera architettura.

```
contract FedAggregatorNFT is ERC721, Ownable {
    string public modelHash;
    string public modelWeightHash;
    uint256 public currentRound;
    uint8 public federatedStatus;
    address public aggregatorAddress;

    mapping(uint256 => string) public roundHashes;
    mapping(uint256 => string) public roundDetails;
    mapping(uint256 => string) public roundWeights;

    //Costruttore
    constructor(address _aggregatorAddress, string memory _modelHash) ERC721("FedAggregatorNFT", "FAN") Ownable(msg.sender) {
        require(_aggregatorAddress != address(0), "Aggregator cannot be zero address");
        require(bytes(_modelHash).length > 0, "Model hash cannot be empty");

        aggregatorAddress = _aggregatorAddress;
        modelHash = _modelHash;
        currentRound = 1;
    }
}
```

Figure 2: Costruttore Contratto FeedAggregator

## FedAggregator

- **Modelhash:** Funzione pubblica, può essere utilizzata da chiunque, restituisce l'hash del Modello
- **getAggregator:** Funzione pubblica, restituisce l'indirizzo dell'aggregatore incaricato del processo FL
- **getRoundDetails:** Funzione external, prende in input il numero del round e restituisce tutte le informazioni (hash dei pesi, metriche) aggregate
- **getCurrentRound:** Funzione external, restituisce il numero del round corrente in uint256
- **federatedStatus:** Funzione pubblica, restituisce lo stato dell'apprendimento federato; 0 se non è concluso 1 altrimenti
- **endFederation:** Funzione external, può essere utilizzata solo dal proprietario del contratto (Aggregatore) e imposta il federatedStatus a 1
- **mint:** Funzione external, può essere utilizzata solo dall'aggregatore quando coincide con il proprietario del contratto e il federatedStatus è 0. Questa funzione prende in input l'hash dei pesi aggregati dopo un round di apprendimento e tutte le informazioni necessarie riguardanti il round in formato stringa, e come risultato genera e assegna un NFT del round eseguito all'aggregatore

## FedPeer

```

contract FedPeerNFT is ERC721, Ownable {

    address public peerAddress;           // Indirizzo del peer
    address public aggregatorAddress;      // Indirizzo del contratto aggregatore
    uint256 public lastParticipatedRound; // Ultimo round partecipato
    uint8 public peerStatus;              // 0 = Attivo, 1 = Inattivo
    address public aggregator;

    mapping(uint256 => string) public roundDetails;

    // Costruttore
    constructor(address _peerAddress, address _aggregatorAddress) ERC721("FedPeerNFT", "FPN") Ownable(msg.sender) {
        require(_peerAddress != address(0), "Peer address cannot be zero");
        require(_aggregatorAddress != address(0), "Aggregator address cannot be zero");

        peerAddress = _peerAddress;
        aggregatorAddress = _aggregatorAddress;
        aggregator = msg.sender;
        peerStatus = 0; // Peer attivo di default
        lastParticipatedRound=0;
    }
}

```

Figure 3: Costruttore Contratto FedPeer

- **getPeerStatus:** Funzione external, restituisce lo stato del peer; 0 se attivo 1 altrimenti
- **getPeerAddress:** Funzione external, restituisce l'indirizzo del peer che usa lo smart contract
- **getAggregatorAddress:** Funzione external, restituisce l'indirizzo dello smart contract FedAggregator
- **getLastParticipatedRound:** Funzione external, restituisce l'ultimo round a cui il peer ha partecipato in uint256
- **mint:** Funzione external, può essere utilizzata solo dal peer quando coincide con il proprietario del contratto e peerStatus è 0. Questa funzione prende in input il numero del round e le informazioni del peer al round corrente, e come risultato genera e assegna l'NFT del round eseguito al Peer
- **stopPeer:** Funzione external, può essere utilizzata solo dall'aggregatore. Questa funzione cambia lo stato del peer a 1
- **restartPeer:** Funzione external, può essere utilizzata solo dall'aggregatore. Questa funzione cambia lo stato del peer a 0
- **getRoundDetails:** Funzione external, prende in input il numero del round e restituisce tutte le informazioni (hash dei pesi, metriche) del singolo peer

### 3.3 Utilizzo di Ganache come TestNet

Per connettere il sistema di training AI con gli smart contract, è stato utilizzato Ganache [13], un ambiente di sviluppo blockchain locale che permette la simulazione di una rete Ethereum privata. Ganache fornisce una serie di funzionalità avanzate, tra cui la generazione di account Ethereum con chiavi private, il mining istantaneo di blocchi per eseguire transazioni in tempo reale e la possibilità di controllare il gas fee senza costi reali. Inoltre, supporta un'interfaccia grafica user-friendly che permette di monitorare in modo intuitivo le transazioni, i contratti distribuiti e gli account coinvolti, come si può apprezzare

in figura 4 . Attraverso questo strumento, ho potuto generare un ambiente di sviluppo controllato, gestendo la creazione e il deployment dei contratti intelligenti e monitorando in dettaglio il flusso delle transazioni necessarie per registrare i dati dell’addestramento AI sulla blockchain. Grazie a questa infrastruttura, è stato possibile testare e validare le interazioni tra i nodi del Federated Learning e la blockchain senza i rischi e i costi associati a una rete pubblica.


CURRENT BLOCK 11	GAS PRICE 20000000000	GAS LIMIT 6721975	HARDFORK MERGE	NETWORK ID 5777	RPC SERVER HTTP://127.0.0.1:7545	MINING STATUS AUTOMINING	WORKSPACE FEDERATED_CHAIN	SWITCH	
TX HASH									CONTRACT CALL
0x63a4fe1ef212fcc470da1cc835418a010265e5ee6a99a74fdb50810c00d9f12									
FROM ADDRESS 0x422120116dc4DA07905c6D04579659b72a3b5e31					TO CONTRACT ADDRESS 0x18D0d370193B2b2bac94F68aE0b3E8E4C4a4b11e		GAS USED 29086	VALUE 0	
TX HASH									CONTRACT CREATION
0xdb187277607a2171e4a6d696c0ee107aa6850b69f23cec066e372d2e8d74cf61									
FROM ADDRESS 0x422120116dc4DA07905c6D04579659b72a3b5e31					CREATED CONTRACT ADDRESS 0x18D0d370193B2b2bac94F68aE0b3E8E4C4a4b11e		GAS USED 2745359	VALUE 0	

Figure 4: GUI Ganache

## 4 Risultati e Conclusioni

Prima di avviare il training, l'account assegnato all'aggregatore sulla blockchain di Ganache ha effettuato il deploy del contratto **FedAggregator** per sé stesso e di cinque contratti **FedPeer**, uno per ciascuno dei cinque account destinati ai peer (figura 5). Successivamente, utilizzando le funzioni messe a disposizione da OpenZeppelin, l'aggregatore ha trasferito la proprietà di questi contratti ai rispettivi peer, decentralizzando il controllo del sistema e rafforzando il concetto di *federated governance*, in cui ogni nodo mantiene autonomia sulle operazioni a lui delegate. Una volta configurato l'ambiente di test, è

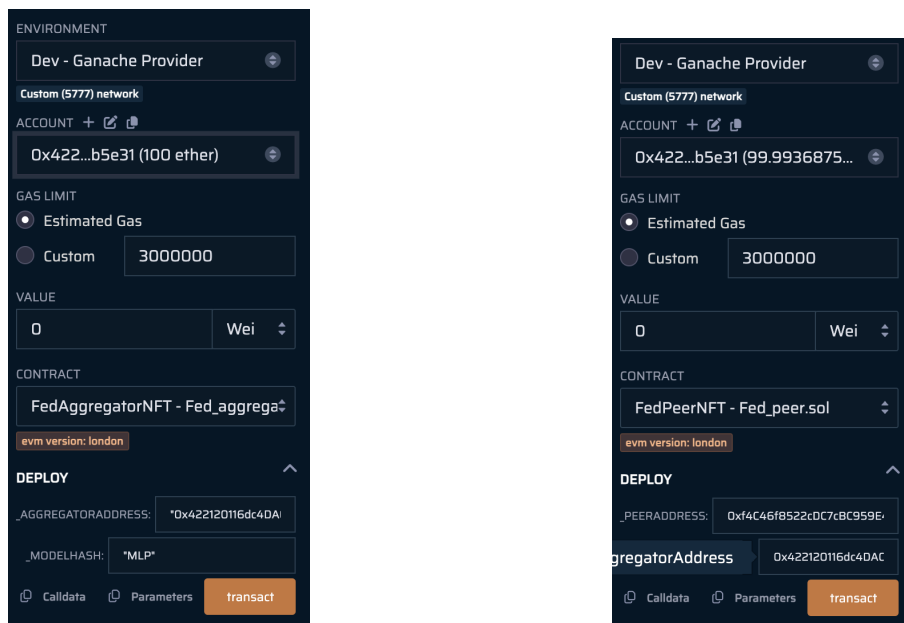


Figure 5: GUI Remix per deploy di contratti

stato avviato il processo di addestramento del modello AI, basato su un'architettura in cui operano due ruoli principali: l'aggregatore e i peer. Ogni peer esegue il training locale sui propri dati e, al termine di ogni round, invia le metriche e gli aggiornamenti del modello al proprio contratto **FedPeer**. Questi contratti registrano le informazioni *on-chain* tramite la funzione **mint**, generando un NFT ERC721 che include i metadati relativi al round di apprendimento. Parallelamente, l'aggregatore raccoglie i parametri aggiornati da tutti i peer, li combina per ottenere una visione globale dello stato del training e utilizza la funzione **mint** del contratto **FedAggregator** per memorizzare *on-chain* il riepilogo aggregato, garantendo trasparenza e tracciabilità dell'intero processo. In figura 6 è possibile osservare l'output del kernel di spyder al completamento del round 9, nel quale sono indicate tutte le transazioni effettuate dai peer con i loro dati, e il blocco in cui è presente la transazione.

L'adozione dello standard ERC721 permette di rappresentare in modo univoco ogni round di addestramento, assicurando l'immutabilità e la verificabilità dei dati. Inoltre, la gestione decentralizzata dei contratti **FedPeer** isola gli aggiornamenti, riducendo il rischio di collusione o manipolazione. Ogni contratto opera come un modulo indipendente, con funzioni standardizzate per la registrazione e il recupero dei parametri locali, facilitando l'interazione con l'aggregatore e garantendo un controllo dettagliato sulle transazioni *on-chain*.



```

--- Federated Round 9/10 ---
169/169 ----- 0s 2ms/step - accuracy: 0.9574 - loss: 0.1475 - val_accuracy: 0.9608 - va
0.1558
169/169 ----- 0s 1ms/step - accuracy: 0.9549 - loss: 0.1567 - val_accuracy: 0.9517 - va
0.1585
169/169 ----- 0s 1ms/step - accuracy: 0.9549 - loss: 0.1583 - val_accuracy: 0.9450 - va
0.2005
169/169 ----- 0s 1ms/step - accuracy: 0.9587 - loss: 0.1435 - val_accuracy: 0.9358 - va
0.2074
169/169 ----- 0s 1ms/step - accuracy: 0.9572 - loss: 0.1465 - val_accuracy: 0.9617 - va
0.1446
313/313 - 0s - 464us/step - accuracy: 0.9590 - loss: 0.1376
Tx inviata per il peer con info: peer=1, round=9,
hash=7d606b3c57bc9b28fba5a96d968005171e4663387226263c2efbee9e3f9c2f70, accuracy=0.9570
Tx hash: a0b7cc59952089f18dd43ff5a4b31447cc8a0d24aeb436b41ddaa42d9894cbd4
Tx inclusa nel blocco: 60
-----
Tx inviata per il peer con info: peer=2, round=9,
hash=e1b855d672de2ef683591724e4a67e2ba240f83fb5b6329a9637b46ffdf98a52, accuracy=0.9540
Tx hash: 16378f0a1cc297c2ef5f6ba13de5ae1317895614dd41abce2a79045956fd0b88
Tx inclusa nel blocco: 61
-----
Tx inviata per il peer con info: peer=3, round=9,
hash=5bfea8f6bb89bc0a9e0e8513f40a4f459634cfc3756e9a8dcf02192039e59502, accuracy=0.9520
Tx hash: a90501d8ec670b65b352a0a734473c3367c9b19e2d02fc561a14f0d12d0e3597
Tx inclusa nel blocco: 62
-----
Tx inviata per il peer con info: peer=4, round=9,
hash=fb1264c3ca387a41f418ed32b0e69abff260a3f1946643ab27a19e4836398ba6, accuracy=0.9475
Tx hash: f21f951f53889c2f764dc388a2af9a3adb12e4396b15ff137e04d8d17d12b937
Tx inclusa nel blocco: 63
-----
Tx inviata per il peer con info: peer=5, round=9,
hash=dedf1a17d92d666b2a0957548f4ef24f5afcc2fd2e80c1ce0d4284ef5c424a27, accuracy=0.9530
Tx hash: a0424c42e27aeb422f9ef1c10aa7a57c9d3d6af3bc492c169923c84da4613665
Tx inclusa nel blocco: 64
-----
Tx inviata per l'aggregatore, hash: f955c8cbfd7366762a346372ad5f68fbc71640ffe3ec16c6210537eb81e21a62
Tx inclusa nel blocco: 65
Status: 1

```

Figure 6: Output e transazioni round 9

Ogni transazione che modifica la blockchain comporta un costo in *gas*, il cui pagamento avviene in **Ether** (ETH). Durante ogni round di addestramento, la blockchain viene aggiornata con la creazione di *sei* NFT:

- un **NFTAggregator**, generato dall'aggregatore, che contiene l'hash dei pesi aggregati e le informazioni sul round;
- cinque **NFTPeer**, ciascuno associato al round e al peer che lo ha prodotto.

Questa operazione comporta un consumo di *gas* variabile a seconda del ruolo. Nella blockchain di *Ganache*, il valore del *gas price* è stato fissato a **2.000.000 di wei**, dove **1 wei** corrisponde a  $10^{-18}$  ETH.

Il costo effettivo di una transazione è calcolato come:

$$\text{Costo} = \text{Gas} \times \text{GasPrice}.$$

Nella Tabella 1 è possibile osservare i costi complessivi per l'aggregatore e per un singolo peer a fine addestramento. L'aggregatore sostiene costi più elevati perché si occupa:

- della generazione dei contratti (costo medio di 2,750,000gas per 6 volte);
- del trasferimento della proprietà ai peer (costo medio di 29,086gas per 5 volte);
- del minting dei propri NFT per ogni round (costo medio di 320,000gas per 10 volte).

Il peer, invece, paga soltanto la funzione di minting, con un costo medio di 180,830gas ripetuto 10 volte.

A fine addestramento, il costo complessivo per l'aggregatore risulta essere di circa **0,4 ETH**, che corrisponde a circa **1040,21 USD** (supponendo  $1\text{ETH} \approx 2,600\text{USD}$ ). Per il singolo peer, invece, il costo complessivo è pari a **0,036 ETH**, ossia circa **93,62 USD**.

È importante sottolineare che tali costi possono variare in base alla rete utilizzata (ad esempio una *testnet*, una *mainnet* o una soluzione *Layer 2*) e al livello di congestione della blockchain nel momento della transazione.

	Deploy	TransferOwnership	Mint
<b>Aggregatore</b>	0.334 ETH	0.0029 ETH	0,063 ETH
<b>Peer</b>	0	0	0,036 ETH

Table 1: Tabella costi per funzione

Ogni volta che un peer o l'aggregatore esegue un'operazione *on-chain*, come il *minting* di un NFT o il trasferimento di proprietà, viene generata una transazione inclusa in un blocco nella blockchain simulata su Ganache. Questa testnet, funzionando come una blockchain locale per il testing, processa queste transazioni in modo quasi istantaneo, producendo blocchi che registrano tutte le operazioni effettuate nel sistema, come possibile osservare in figura 7. Ogni blocco è crittograficamente collegato al precedente, garantendo l'integrità della catena e la non modificabilità dei dati già registrati. Questo meccanismo permette di mantenere una cronologia trasparente di tutti gli aggiornamenti dei modelli AI, fornendo un *audit trail* verificabile e immutabile dell'intero processo di *federated learning*.

Sebbene il sistema sia stato testato su Ganache, l'architettura è stata progettata per essere scalabile e facilmente adattabile a reti pubbliche o soluzioni *Layer2*. Questa flessibilità consente di ottimizzare il bilanciamento tra costi di transazione, velocità di esecuzione e sicurezza, permettendo una futura integrazione in ambienti di produzione reali. La registrazione *on-chain* di ogni fase dell'addestramento assicura infine un meccanismo di *auditing* trasparente e decentralizzato, offrendo la possibilità di verificare l'intero processo in qualsiasi momento.

BLOCK <b>71</b>	MINED ON 2025-02-16 16:12:57	GAS USED 313253	1 TRANSACTION
BLOCK <b>70</b>	MINED ON 2025-02-16 16:12:57	GAS USED 180842	1 TRANSACTION
BLOCK <b>69</b>	MINED ON 2025-02-16 16:12:57	GAS USED 180842	1 TRANSACTION

Figure 7: GUI Ganache

Una volta terminato il processo di addestramento, i dati immutabili relativi a ogni round sono stati registrati nella blockchain. Questi possono essere *auditati* in modo estremamente semplice utilizzando la GUI di Remix: è sufficiente individuare la funzione `GetRoundDetails` nei contratti `FedPeer` o `FedAggregator`, inserire il numero del round di interesse nell'apposito campo e, in risposta, Remix fornirà i dettagli dell'addestramento riferiti a quel round. Per ottenere ulteriori informazioni, l'auditor può inoltre utilizzare altre funzioni pubbliche – come `currentRound` o `getAggregator` – che permettono di verificare lo stato attuale dell'addestramento o individuare l'indirizzo del contratto *aggregator*, fornendo una visione più chiara dell'intero processo. È importante notare che

tali operazioni di lettura (ossia le chiamate di consultazione dei dati *on-chain*) non comportano alcun costo, poiché non modificano la struttura della blockchain.

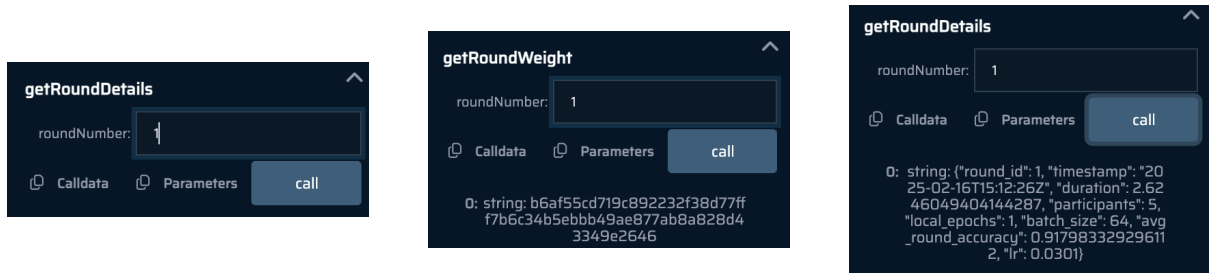


Figure 8: GUI Remix per l'utilizzo di funzioni

## 5 Contributo personale

In questo lavoro di tesi è stato affrontato il problema della sicurezza e privacy dei dati AI, analizzando le vulnerabilità della gestione centralizzata. Si è individuato nel Federated Learning un metodo efficace per proteggere i dati senza condividerli direttamente. Quindi è stata progettata un'architettura Web3 integrata con la blockchain e sviluppato due smart contract (FedAggregator e FedPeer) per registrare i parametri di apprendimento e decentralizzare il controllo. Dopo aver adattato il codice AI per interfacciarsi con i contratti, è stata eseguita una simulazione del sistema su Ganache, verificandone la correttezza e la tracciabilità. I risultati ottenuti confermano la fattibilità di un approccio decentralizzato al Federated Learning e la sua applicabilità su reti pubbliche o Layer2.

## References

- [1] EU, “Eu artificial intelligence act,” *available online : <https://artificialintelligenceact.eu/>*, last accessed 18/02/2025.
- [2] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, “Communication-efficient learning of deep networks from decentralized data,” in *Artificial intelligence and statistics*, PMLR, 2017, pp. 1273–1282.
- [3] S. Nakamoto, “Bitcoin whitepaper,” *URL: <https://bitcoin.org/bitcoin.pdf>* (: 17.07. 2019), vol. 9, p. 15, 2008.
- [4] Z. Liu, Y. Xiang, J. Shi, *et al.*, “Make web3. 0 connected,” *IEEE transactions on dependable and secure computing*, vol. 19, no. 5, pp. 2965–2981, 2021.
- [5] Z. Zheng, S. Xie, H.-N. Dai, *et al.*, “An overview on smart contracts: Challenges, advances and platforms,” *Future Generation Computer Systems*, vol. 105, pp. 475–491, 2020.
- [6] W. Cai, Z. Wang, J. B. Ernst, Z. Hong, C. Feng, and V. C. Leung, “Decentralized applications: The blockchain-empowered software system,” *IEEE access*, vol. 6, pp. 53 019–53 033, 2018.
- [7] L. Petrosino, L. Masi, F. D’Antoni, M. Merone, and L. Vollero, “A zero-knowledge proof federated learning on dlt for healthcare data,” *Journal of Parallel and Distributed Computing*, vol. 196, p. 104 992, 2025.
- [8] L. Deng, “The mnist database of handwritten digit images for machine learning research [best of the web],” *IEEE signal processing magazine*, vol. 29, no. 6, pp. 141–142, 2012.
- [9] E. Community, “Remix ethereum ide,” *available online : <https://remix.ethereum.org/lang=enoptimv0.8.26+commit.8a97fa7a.js>*, last accessed 14/02/2025.
- [10] Z. G. Ltd, “Openzeppelin,” *available online : <https://www.openzeppelin.com/>*, last accessed 18/02/2025.
- [11] E. William, S. Dieter, E. Jacob, and S. Nastassia, “Erc721: Non fungible token standard,” *available online : <https://eips.ethereum.org/EIPS/eip721>*., Ethereum Improvement Proposals, no. 721, January 2018.
- [12] V. Buterin *et al.*, “Ethereum white paper,” *GitHub repository*, vol. 1, pp. 22–23, 2013.
- [13] T. Suite, “Ganache,” *available online : <https://archive.trufflesuite.com/docs/ganache/>*, last accessed 18/02/2025.

# Ringraziamenti

Desidero esprimere la mia sincera gratitudine a tutte le persone che hanno contribuito, in modi diversi, al completamento di questa tesi.

Un ringraziamento speciale va al mio correlatore, **Lorenzo Petrosino**, per il supporto, la guida preziosa e i consigli che hanno reso possibile questo lavoro. La sua competenza e disponibilità sono stati fondamentali per affrontare le sfide di questa ricerca.

Un grazie di cuore alla mia famiglia, e in particolare a **mamma e papà**. Grazie per avermi permesso tutto questo, per avermi sostenuto nei momenti più duri, per avermi insegnato tutto ciò che mi è servito per arrivare fino a qui. Il vostro amore, la vostra pazienza e la vostra fiducia sono stati il mio più grande punto di forza.

Un pensiero speciale ai miei amici e colleghi. Ringrazio tutti i membri dello **Sciaino**, per aver condiviso con me momenti di confronto, studio e crescita personale. La vostra presenza ha reso questo viaggio più stimolante e meno faticoso. Grazie a tutti voi che mi avete accompagnato in questo duro percorso: con voi sono cresciuto, ho affrontato il passaggio da ragazzo a uomo.

Un ringraziamento particolare agli **Squali**, per aver mantenuto la mia sanità mentale nel corso degli anni e per la costante motivazione che siete stati. Il vostro supporto e la vostra energia hanno reso ogni sfida più affrontabile e ogni momento più speciale.

Desidero inoltre ringraziare particolarmente **Josephine**, che negli ultimi tre mesi mi ha spronato a fare meglio e mi è stata vicina con il suo sostegno e la sua incrollabile fiducia. La sua presenza è stata una fonte di motivazione e forza che ha reso questo percorso ancora più significativo.

Infine, un grazie a chiunque abbia contribuito, anche indirettamente, alla realizzazione di questo lavoro, offrendomi ispirazione, supporto tecnico o semplicemente un sorriso nei momenti più impegnativi.

Questa tesi è il frutto di un percorso che non avrei potuto affrontare da solo. A tutti voi, **grazie di cuore**.

## Forza Lazio!