Multimedia Tools and Applications (2022) 81:25029–25050
https://doi.org/10.1007/s11042-022-12900-5

Privacy-preserving federated learning for scalable and high data quality computational-intelligence-as-a-service in Society 5.0

Amirhossein Peyvandi[1] · Babak Majidi[1,2] · Soodeh Peyvandi[3] · Jagdish C. Patra[4]

Received: 11 September 2020 / Revised: 8 February 2022 / Accepted: 9 March 2022 /
Published online: 22 March 2022

Abstract
Training supervised machine learning models like deep learning requires high-quality labelled datasets that contain enough samples from various categories and specific cases. The Data as a Service (DaaS) can provide this high-quality data for training efficient machine learning models. However, the issue of privacy can minimize the participation of the data owners in DaaS provision. In this paper, a blockchain-based decentralized federated learning framework for secure, scalable, and privacy-preserving computational intelligence, called Decentralized Computational Intelligence as a Service (DCIaaS), is proposed. The proposed framework is able to improve data quality, computational intelligence quality, data equality, and computational intelligence equality for complex machine learning tasks. The proposed framework uses the blockchain network for secure decentralized transfer and sharing of data and machine learning models on the cloud. As a case study for multimedia applications, the performance of DCIaaS framework for biomedical image classification and hazardous litter management is analysed. Experimental results show an increase in the accuracy of the models trained using the proposed framework compared to decentralized training. The proposed framework addresses the issue of privacy-preserving in DaaS using the distributed ledger technology and acts as a platform for crowdsourcing the training process of machine learning models.

Keywords Federated learning · Blockchain · Privacy preserving · Decentralized machine learning · Data as a service · Society 5.0

✉ Babak Majidi
b.majidi@khatam.ac.ir

[1] Department of Computer Engineering, Khatam University, Tehran, Iran

[2] Emergency and Rapid Response Simulation (ADERSIM) Artificial Intelligence Group, Faculty of Liberal Arts & Professional Studies, York University, Toronto, Canada

[3] Process Management & Business Intelligence, University of Applied Sciences Upper Austria, Steyr, Austria

[4] Faculty of Science, Engineering and Technology, Swinburne University of Technology, Melbourne, Australia

1 Introduction

The issue of the privacy is one the most important issues that should be addressed in Society 5.0. It can be estimated that 2.5 quintillion bytes (25×10^5 TB) of data is generated and accessed daily [29]. Although all of this data is not useful for machine learning, if only 1% of this data is valuable and can be utilized in smart environment modelling and applications, it would amount to 25,000 TB of usable data generated daily. Using this data to train machine learning models allows researchers to enhance computational intelligence solutions for different industries, healthcare, management, business, smart environments, and many other applications in Society 5.0. However, an essential issue for using this data is preserving the privacy of people who generate or own the data. There has been a significant discussion regarding the issue of preserving the privacy of data providers. A solution for this problem is decentralized transmission and storage of data as opposed to the centralized approach in which data is stored in a server resulting in open opportunities for various attacks.

A decentralized solution for training machine learning models is federated learning [38]. This method trains the model across multiple nodes or servers that hold local data samples. However, this method does not necessarily protect the privacy of data providers [25]. Blockchain technology can be used as a solution to address this issue. In this paper, a blockchain-based federated learning framework for secure, scalable, and privacy-preserving machine learning, which combines blockchain-based Data as a Service (DaaS) and Machine Learning as a Service (MLaaS), called Decentralized Computational Intelligence as a Service (DCIaaS), is proposed. In the proposed framework, the models will be trained off-chain on the data provider's side, and only the learned model parameters and weights will be shared on the blockchain. The decentralized DaaS allows the data providers to make the data available to the machine learning specialists based on their demand. The provision of the data is irrespective of geospatial or other relations of the data provider and data consumer. In the proposed framework, each data point is coupled with its owner, and data remains distributed and private.

Training supervised machine learning models like deep learning requires high-quality labelled datasets which contain enough samples from each category and specific cases. The proposed framework helps in the creation of better machine learning datasets by increasing samples of minority classes. The main contributions of this manuscript are as follows: 1. A computationally inexpensive framework that combines blockchain and federated learning allowing privacy-preserving MLaaS; 2. The proposed framework can increase the accuracy of machine learning models in comparison with decentralized training; 3. Two practical multimedia case studies that demonstrate the applications of the proposed framework for Society 5.0.

The rest of the paper is organized as follows: In Section 2, a review of recent literature related to federated learning, privacy-preserving, and machine learning on the blockchain is presented. In Section 3, the proposed DCIaaS framework is presented and detailed. In Section 4, experimental results and practical applications of the proposed framework are discussed. Finally, Section 5 concludes the paper and provides future directions for the research.

2 Related works

Recently, federated learning has been widely investigated for various applications in the Internet of Things (IoT) [33, 70], Internet of Medical Things (IoMT), and Industry 4.0 [3,

67]. During the COVID-19 pandemic, it has been proposed for improved COVID-19 detection [68]. It has also been utilized for the secure classification of COVID-19 chest X-ray images [39]. Moreover, blockchain technology is also proposed as a reliable tool for secure Biomedical Data as a Service (BDaaS) for epidemic management [51]. Furthermore, federated learning has been utilized for training models on distributed data located in different medical institutions [55] . Rajendran et al. [53] proposed cloud-based federated learning. This method uses a centralized approach and therefore, even though a 3% increase in performance of the trained models is reported, it can lead to exposure of sensitive medical records. In most federated learning frameworks, the aggregation of trained models takes place on a centralized server, and the shared weights and models are open to attacks. Ge et al. [20] propose a privacy-preserving medical framework which utilizes federated learning for health purposes. In this research, the sharing of trained models and weights does not take place on a secure channel.

One of the main issues with federated learning is that it is not secure. For example, the participants may behave maliciously during gradient collection or parameter updating process, and the server may act maliciously as well. The worst-case scenario is when federated learning is utilized in a centralized setting, i.e., storing all of the data and parameters in a single server, which multiplies risk factors. The decentralized federated learning is also vulnerable because even one single malicious server can pose a threat to the data and models. There are researches which prove that the intermediate gradients can be used to infer important information about the training data [44, 59]. Hitaj et al. [25] demonstrate that a federated deep learning approach does not protect the training data. They developed an attack which exploits the real-time nature of the learning process and allows the adversary to train a Generative Adversarial Network (GAN), which generates prototypical samples of the targeted training set that was meant to be private. Moreover, they demonstrate that record-level differential privacy applied to the shared parameters of the model is ineffective. Moreover, other researchers questioned the security of federated learning [57]. There are computationally expensive solutions proposed to address this problem. Phong et al. [52] presented a privacy-preserving deep learning system in which different learning participants perform deep learning over a combined dataset without revealing the participants' local data to a central server. In this work, the authors connect deep learning and cryptography and utilize asynchronous stochastic gradient descent in combination with homomorphic encryption.

In order to solve the problems of federated learning, machine learning on a blockchain can be used. DeepChain [63] is a distributed and secure deep learning framework which aims to solve the aforementioned problems. DeepChain provides a value-driven incentive mechanism based on blockchain in order to make the participants behave correctly. Moreover, their proposed framework guarantees the privacy of participants and data providers during the training process. This framework preserves the privacy of local gradients and guarantees the auditability of the training process. In DeepChain, two smart contracts are utilized. One contract is dedicated to the management of data providers, while the other contract controls "workers" who train the models. By utilizing blockchain, the authors ensure that no malicious activity can happen. Goel et al. [22] proposed that by utilizing a cryptographic hash, as well as symmetric/asymmetric encryption and decryption algorithms, security will be ensured without any centralized authority. The authors proposed DeepRing, which utilizes the learned parameters of a standard deep neural network model and is secured from external adversaries by cryptography and blockchain technology. Their proposed framework transforms each layer of the deep neural network into a block and handles them accordingly. Baldominos et al. [6] proposed a blockchain-based system named "Coin.AI," in which the mining arrangement

requires training deep learning models, and a block is only mined when the performance of a model exceeds a threshold. The distributed system allows the blockchain nodes to verify the models delivered by miners, determining when a block is to be generated. Moreover, the authors introduced a proof-of-storage scheme for rewarding users that provide storage for the deep learning models. Fadaeddini et al. [18] proposed a secure decentralized peer-to-peer framework in order to train deep neural network models on distributed ledger technology on Stellar blockchain [27]. A Deep Learning Coin (DLC) is proposed for blockchain compensation. In order to address the issue of data sharing for platforms that depend on a Trusted Third Party (TTP), Naz et al. [47] proposed a blockchain-based secure data and file sharing platform by utilizing IPFS and smart contract technology. In this method, the owner first uploads metadata, which is then divided into n secret shares. Furthermore, customers can review files and comment on them. Addressing the issue of privacy at the data level is another solution for privacy-preserving, which is investigated by Hajiabbasi et al. [1].

Comparison of the proposed DCIaaS framework with the above-mentioned machine learning on blockchain proposals shows that the proposed DCIaaS framework does not have the vulnerabilities of federated learning-based approaches such as [13, 48], that may expose sensitive and personal data during their training process. On the other hand, although works such as [22, 63] aim to eliminate problems of federated learning by utilizing blockchain technology since they are built around distributed systems and the training data is still shared between miners or other individuals, this distribution of models and data can still prove to be harmful. However, the proposed DCIaaS framework solves this issue as data holders are not required to share their data. In DCIaaS, models are trained on the data owner's end of the blockchain and only the trained weights are shared in a secure manner. Furthermore, the proposed framework does not require the expensive computations of cryptographic-based methods. Further comparison of both results and the advantages of the proposed framework is discussed in the experimental results section of the manuscript.

3 Decentralized computational-intelligence-as-a-service

The proposed DCIaaS uses decentralized DaaS, in which the data remains distributed on the data owners' nodes, and combined with blockchain and federated learning, it remains anonymised. This is opposed to the centralized approach in which data is aggregated on a central server, and at best, the data can be pseudonymized. It should be noted that training models on blockchain infrastructure is hard and consumes a significant amount of time, money, and resources. Therefore, in the proposed framework, the actual training process is performed off-chain. In the proposed DCIaaS framework, there are three groups of primary nodes. The first nodes are Data Providers such as governmental bodies, organizations, companies, hospitals, medical centres, citizens, and other researchers. This group is not required to share their data. Instead, they are only required to offer a sample of the data (i.e., a few records of their dataset), plus a description of the dataset and its features. Moreover, if this sample contains any sensitive information, the data providers can either anonymise the sample by completely removing such attributes and only provide a detailed description of them or use pseudonymization techniques. The data providers are the *clients* in the federated learning algorithm.

The second nodes are referred to as Applicants. This group will not have direct access to sensitive data and will only share their algorithms and codes with the Data Providers. They can

implement their algorithms and codes by analysing the shared sample and its description. Moreover, when signing a contract with data providers, they are required to send their overall proposal to the data providers. This proposal consists of a summary of what their algorithm is going to do, what programming language and which frameworks and libraries are used, and required resources, such as CPU, TPU, and GPU. The final node is the Smart Contract. In the proposed framework, one smart contract [41], called TrainingModel, is utilized. This smart contract is used to control the contracts signed between a Data Provider and an Applicant. Furthermore, it is responsible for performing federated learning (applying federated learning algorithm to model weights) on the blockchain. The model for the relationship between Data Providers (*clients*) and Applicants is presented in Fig. 1.

In this section, first, the issue of privacy-preserving in decentralized machine learning is discussed, and then the federated learning algorithm used in DCIaaS is presented. Then, the details of the implementation of DCIaaS on the blockchain are presented.

3.1 Preservation of privacy for decentralized machine learning

Due to the high computational and resource cost of deep learning algorithms, data scientists often rely upon MLaaS to outsource the computational load onto third-party servers. However, outsourcing the computation raises privacy concerns when dealing with sensitive information, such as medical records. Furthermore, privacy regulations like the European GDPR, limit the collection, distribution, and use of sensitive data and information. Recent advances in privacy-preserving techniques, such as Homomorphic Encryption (HE), federated learning, and Differential Privacy (DP), have enabled model training and inference over protected data. These data privacy techniques aim at reducing the amount of sensitive information that data carry. Overall, MLaaS relies on three different types of privacy requirements [14]:

1- *Input privacy*, which aims to preserve data privacy during training or inference. This requirement is needed when the data is sent to an external, non-trusted party (a cloud server) that performs the computation;

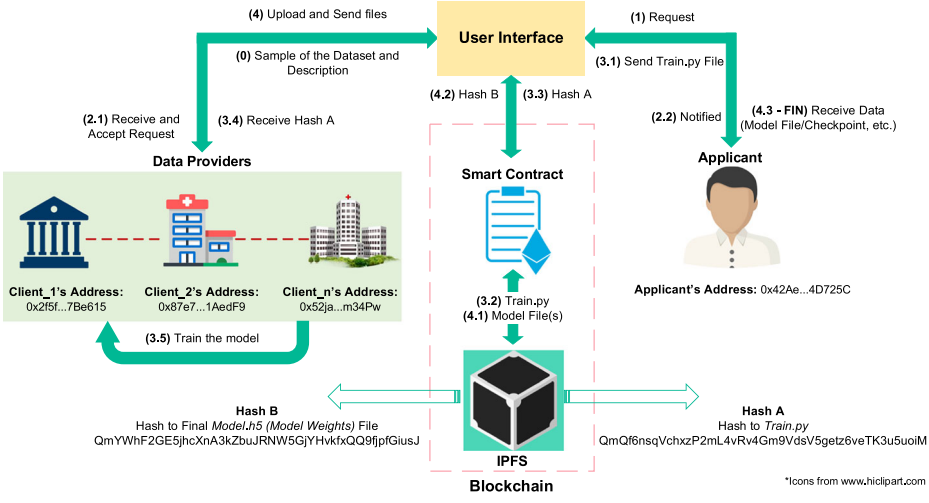


Fig. 1 DCIaaS for privacy preserving federated learning

2- *Output privacy*, that ensures the non-revelation of private information about the data from the products of the training (i.e., the model) or inference (i.e., the output predictions);
3- *Model privacy* is the property ensuring the non-revelation of the attributes that define a model, such as architecture and weights.

There are three main solutions for addressing these issues. The first solution is HE that is an encryption scheme. Homomorphism is a mathematical concept whereby the structure is preserved throughout a computation. Since only certain mathematical operations, such as addition and multiplication, are homomorphic, the application of HE to neural networks requires the procedures defined within the algorithm to conform to these limitations [28]. In order to implement HE and encrypt the models' weights, the *ncryption* scheme [11, 36] can be utilized. This method takes the secret key with large noise as input and outputs unencrypted data of the same input with a fixed amount of noise. Let R be the unencrypted matrix data of the mini-batch dataset with the size of $N \times M$. Before the encryption of a tensor, a private key matrix φ with size $N \times N$ as:

$$\begin{bmatrix} \varphi_{11} & \cdots & \varphi_{1N} \\ \vdots & \ddots & \vdots \\ \varphi_{N1} & \cdots & \varphi_{NN} \end{bmatrix} \tag{1}$$

is created. This key is only accessed by the participants who are authorized and shared the mini-batch dataset, with $\mathbb{R}_{(N)}$ being the plaintext space:

$$\begin{bmatrix} \mathbb{R}_{(1)} \\ \mathbb{R}_{(2)} \\ \vdots \\ \mathbb{R}_{(N)} \end{bmatrix} = \begin{bmatrix} \varphi_{11} & \cdots & \varphi_{1N} \\ \vdots & \ddots & \vdots \\ \varphi_{N1} & \cdots & \varphi_{NN} \end{bmatrix} \otimes \begin{bmatrix} R_{(1)} \\ R_{(2)} \\ \vdots \\ R_{(M)} \end{bmatrix} \tag{2}$$

where $R_{(j)}$ shows the vector data of the j_{th} node of the ledger. The $\otimes$ operator shows the product between two ciphertext:

$$\mathbb{R}_{(j)} = \varphi_{j1}R_1 + \varphi_{j2}R_2 + \ldots + \varphi_{jM}R_N \tag{3}$$

The second solution for addressing the issue of privacy is DP [16]. DP mechanisms often rely on adding noise to the data, which ends up reducing its expressiveness. A differentially private mechanism acting on very similar datasets will return results which are statistically indiscernible. Given privacy mechanism M, which maps inputs from domain D to outputs in the range R, by multiplicative factor ϵ, regardless of the presence or absence of a single individual in two neighbouring datasets d and d' drawn from D, it is probable that for any subset of outputs $S \subseteq R$:

$$Pr[M(d) \in S] \leq e^{\epsilon} Pr\left[M\left(d'\right) \in S\right], \tag{4}$$

where d and d' are correspondent with the same output. This method protects individuals from being identified within the dataset [12]. DP is an example of a perturbative privacy-preserving method, as the privacy guarantee is achieved by the addition of noise to the true output. This noise is usually drawn from a Laplacian distribution, but it can also be drawn from an exponential distribution or via the novel staircase mechanism that provides greater utility compared to Laplacian noise for the same ϵ. The aforementioned description of differential

privacy is often known as ϵ-differential privacy (ϵ-DP). The amount of noise needed for ϵ-DP is controlled by ϵ, and the sensitivity of the function Q defined by:

$$\Delta Q = \max\left(\left\|Q(d)-Q(d)^{'}\right\|_1\right). \tag{5}$$

This maximum is evaluated over all neighbouring datasets in the set D. The output of the mechanism using noise drawn from the Laplacian distribution L is:

$$M(d) = Q(d) + L\left(0, \frac{\Delta Q}{\varepsilon}\right). \tag{6}$$

Moreover, a moderate version of DP known as (ϵ, δ)-DP [16] provides greater flexibility in designing privacy preserving mechanisms and greater resistance to attacks [30]:

$$Pr[M(d)\in S] \leq e^{\epsilon} Pr\left[M\left(d^{'}\right)\in S\right] + \delta. \tag{7}$$

The Gaussian mechanism is commonly used to add noise to satisfy (ϵ, δ)-DP, but instead of the *l1* norm, the noise is scaled to the *l2* norm [71]:

$$\Delta_2 Q = \max\left(\left\|Q(d)-Q(d)^{'}\right\|_2\right). \tag{8}$$

Given ϵ, $\delta \in (0, 1)$, the following mechanism satisfies (ϵ, δ)-DP [16]:

$$M(d) = Q(d) + \frac{\Delta_2 Q}{\epsilon}\mathcal{N}\left(0, 2\ln\left(\frac{1.25}{\delta}\right)\right). \tag{9}$$

In this paper, federated learning [63] combined with blockchain is used to address the issue of privacy preservation in MLaaS. Federated learning is a machine learning framework in which multiple parties upload local gradients to a server or multiple servers, and these servers update model parameters with the collected gradients. Moreover, federated learning allows a model to be collaboratively trained using local data from distributed entities without revealing it to the other parties [34]. The clients use their local data to train a local version of the model to compute the updates. Next, these updates are sent back to a central server [56], which aggregates them into a global model. Table 1 shows the strengths and weaknesses of federated learning in comparison with HE and DP.

By itself, federated learning suffers from security issues since the generated model and gradients are shared, and they may be abused to breach privacy. While federated learning is flexible and resolves data governance and ownership issues, it does not guarantee security and

Table 1 Comparison of federated learning with HE and DP

Method	Privacy Goal				Strengths	Weaknesses
	Identity	Dataset	Model	Input		
DP	✓	✓	✓	×	Provable guarantee of privacy	Compromising accuracy of DL model
HE	✓	✓	×	✓	Computing on encrypted data	High computation costs, works on numerical data
FL	✓	✓	✓	×	Decentralized training	High communication cost, high availability

confidentiality by itself unless combined with other methods. A lack of encryption can enable attackers to steal sensitive identifiable information directly from the nodes or interfere with the communication process. The required secure communication can be expensive for large machine learning models or large data volumes. Therefore, federated learning is often combined with other techniques such as HE or DP to preserve input and output privacy. However, these methods are computationally expensive.

In the proposed DCIaaS, the blockchain is utilized for the aggregation part of federated learning. In this case, a smart contract plays the role of "central server" and the privacy-preserving and security are highly guaranteed. Moreover, communication through transactions of the blockchain offers a safe and secure communication channel for sharing weights between different clients and data owners. In the proposed framework, an Ethereum-based [17] smart contract is utilized. The hash function plays the fundamental role in security structure of blockchain over Ethereum network. The hash function used in Ethereum is Keccak-256 [31]. The hash functions compress the volume of data with arbitrary size to the fixed-length. In (10), let $H(x)$ be a hash function (one way – $\{0,\ 1\}^* \rightarrow \{0,\ 1\}^n$) in which x is a random finite length bit-string that produces output Y with fixed length size [45]:

$$\begin{cases} \{0,1\}^* \rightarrow \{0,1\}^n \\ Y = H(x), \end{cases} \tag{10}$$

The cryptographic hash function has three key properties:

1. *Preimage resistance:* Given output Y, it is computationally impractical to find input x;
2. *Second preimage resistance:* Given input x_1 which holds $Y = H(x_1)$, it is difficult to find x_2 such that it yields $(x_1) = H(x_2)$;
3. *Collision resistance:* Given two different inputs x_1 and x_2 $(x_1 \neq x_2)$, it is difficult to get the same output Y: $H(x_1) \neq H(x_2)$.

Therefore, it is difficult to interfere with blocks and transactions in this network. The base layer of Ethereum is its Peer-to-Peer (P2P) network architecture. In a P2P network, each workstation or *node* has the same privileges and responsibilities for sharing, maintaining, and utilizing resources. Furthermore, each workstation can have restrictions upon itself and control privacy and anonymity. The P2P has no dedicated central server, thus making the network decentralized. It has a flat topology, and each node can serve as both server and client at the same time. In Ethereum, the architecture is an overlay network in which nodes logically connect through the Internet. Ethereum uses DevP2P multiprotocol P2P network and extended it by Whisper protocol in order to provide P2P secure communication and Swarm protocol to provide distributed storage.

3.2 The blockchain based federated learning for DCIaaS

The implemented federated learning in this paper is based on McMahan [42] federated learning. Assume that there is a fixed set of K clients, each with a fixed local dataset. At the beginning of each round, a random fraction C of clients is selected, and the server, which in DCIaaS is the Ethereum-based smart contract, sends the current global algorithm state (weights and the current model parameters) to each of these clients. Each client then performs local computation based on the global state and its local dataset and sends an update to the

smart contract. The smart contract then applies these updates to its global state, and the process repeats. The algorithm is applicable to any finite-sum objective of the form:

$$\min_{w\in\mathbb{R}^d} f(w) \qquad where \qquad f(\omega)\stackrel{def}{=}\frac{1}{n}\sum_{i=1}^{n} f_i(w) \tag{11}$$

in which for a deep learning problem, we take $f_i(w) = \ell(x_i, y_i; w)$, meaning that loss of the prediction on example (x_i, y_i) calculates the model parameter w. Assume that there are K clients over which the data is partitioned, with $\mathcal{P}_k$ being the set of indexes of data points on client K, with $n_k = |\mathcal{P}_k|$. Therefore, the previously discussed objective can be rewritten in the form of:

$$f(w) = \sum_{k=1}^{k} \frac{n_k}{n} F_k(w) \qquad where \qquad F_k(w) = \frac{1}{n_k}\sum_{i\in\mathcal{P}_k} f_i(\omega)\cdot \tag{12}$$

If the partition $\mathcal{P}_k$ were formed by distributing the training examples over the clients uniformly at random, we would have $\mathbb{E}_{\mathcal{P}_k}[F_k(w)] = f(w)$. The above equation presents the Federated Averaging (FA) algorithm (the commonly used algorithm for federated learning), which is utilized in this paper. Here, the weight parameters for each client based on the loss values recorded across every data point are being estimated. The FA Algorithm (*FedAvg*) is illustrated in Fig. 2.

This algorithm is consisted of two main loops. Figure 3 shows the implementation of this algorithm using smart contracts in DCIaaS. The red dashed lines in Fig. 3 show which steps are performed using the smart contract in a secure manner. After compiling and getting the initial weights of the designed model, the applicant shares these weights via smart contract. The IPFS can also be utilized if the weights are stored as a file and not a list of numbers or a tensor. Then, the first client will receive the initial global weights and set its local model's weights to the global weights. Next, this client will train the model on its local data, scale the acquired local weights, and adds it to a list. This process will continue for each participant, and a list of aggregated scaled weight will be generated. This is the end of a single Local Iteration (LI). Next, this list of scaled weights of one LI will be sent to the smart contract in order to perform the final step of federated averaging and update the global model. This will be the end of a Global Iteration (GI). In the end, the applicant will receive these finalized weights.

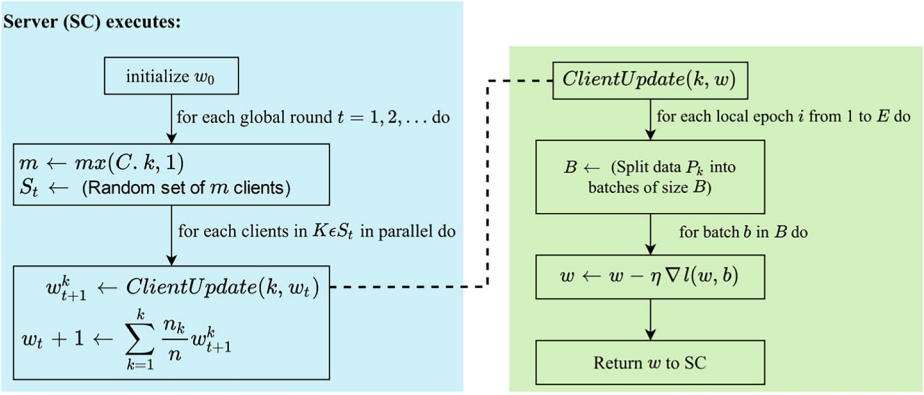


Fig. 2 FA (*FedAvg*) Algorithm

If we assume a scenario in which an Applicant wants to train a model on data available from Data Providers (*clients*), first, data providers are required to provide a sample of their dataset, along with its description. This provides enough information such as its type, format, and size for the node that wants to train a model on the dataset. Next, an applicant connects to the smart contract (*TrainingModel* contract) via the DCIaaS web application. Then, in order to sign a contract with a data provider and send a request for training his model on their dataset, the applicant must provide a proposal of what he intends to do with the data and which programming language, library, frameworks, and resources are needed in order to train his model.

After that, the data provider will receive the request, and after accepting, the applicant will be notified. If the request is accepted, the applicant must upload his code and algorithm on IPFS. Then, he will be given a hash to this file, and the hash will be shared via the smart contract, and the data provider will receive it. At this step, the initial weights of the global model should also be shared with *clients*. Then, data providers (*clients*) will train the model, and after the model is trained, the data provider will upload all files and checkpoints on IPFS and send the hash to the applicant. The applicant will receive this hash and download the model file(s).

The last part of the proposed DCIaaS is the participants' compensation mechanism. Although in the proposed framework, the data remains private, as the training process consumes computing resources, some clients may not be willing to participate. By introducing an encouragement mechanism in which the participants and data providers are rewarded based on their contributions, the participation rate can be improved, and more data providers might be encouraged to join the framework. This can be achieved by combining the Multi-KRUM [9, 54, 69] and the reputation-based incentive protocols [65], in which an encouragement mechanism is designed which prevents the poisoning attack and also rewards participants properly.

In our scenario, after the local model's weights and updates are sent to the smart contract, verifiers calculate the reputation using the Multi-KRUM algorithm and eliminate dubious updates. The verifiers, which are selected based on the VRF [21] from miners, will remove

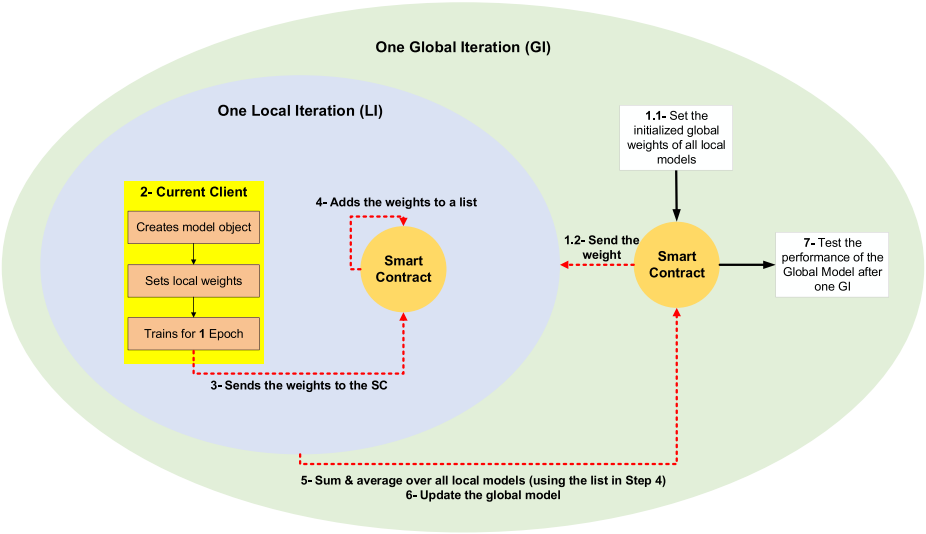


Fig. 3 The illustration of LI and GI of the proposed DCIaaS framework

malicious updates by executing the Multi-KRUM algorithm on updates in the received pool and accept the top majority of the updates received every GI. The verifier will add up Euclidean distances of each client c's updates to the closest $R - f - 2$ updates and denote the sum as each client c's score $S(c)$, where R is the number of updates, and f is the number of Byzantine clients [69]:

$$S_{(c)} = \sum_{c \to k} \|\Delta w_c - \Delta w_k\|^2, \tag{13}$$

where Δ_W is the model update, and $c \to k$ indicate that Δw_k belongs to the $R - f - 2$ nearest updates to Δw_c. The $R - f$ clients who gets the lowest scores will be chosen while rejecting the rest of the clients. The value of the reward is proportional to the client's reputation, meaning if a client's update is accepted by verifiers, the value of reputation is increased by one, and otherwise, it is decreased by one. Each participant is assigned with an initial reputation value γ. The γ is an integer selected form the $set(0, 1, \ldots, \gamma^{max})$, where γ^{max} indicates the highest reputation.

Let h denote the average reputation of all clients. If a miner verifies that a solution is correct and provides a positive evaluation, the reputation of the current client will be increased and stored on the blockchain. If a denotes the evaluation function's output, then $a = H$ indicates a high evaluation, while $a = L$ indicates a low result. The update rule of the reputation γ is as follows:

$$\gamma = \begin{cases} \min(\gamma^{max}, \gamma + 1), & \text{if } a = H \text{ and } \gamma \geq h \\ \gamma - 1, & \text{if } a = L \text{ and } \gamma \geq h + 1 \\ 0, & \text{if } a = L \text{ and } \gamma = h \\ \gamma + 1, & \text{if } \gamma < h \end{cases} \tag{14}$$

where h is the threshold of the selected social strategy. If a client's reputation is h and receives an L (low) feedback after the evaluation, this client's reputation will be decreased to 0, and the status of the reputation will be stored on the blockchain.

3.3 DClaaS software implementation

The smart contract (*TrainingModel*) manages contracts between data providers and applicants. However, before connecting to the smart contract, data providers and applicants are required to set up a crypto wallet or blockchain browser. In doing so, they will be given a unique address, which will be used to identify them in the smart contract. Moreover, in order to connect to this address in other applications, browsers, or wallets, they are given a private key or mnemonic phrase, and as long as they keep this key or phrase safe and do not share it with anyone else, the stakeholders will be safe.

In this paper, a gateway to blockchain called MetaMask browser extension [46] is used for this task and connecting to the smart contract. MetaMask allows connectivity to the distributed web, and instead of running the full Ethereum node, it runs Ethereum decentralized applications in the browser. It should be mentioned that no additional personal information is required, and no information will be stored on a blockchain, and therefore the anonymity of users will be preserved. As for the private records, by controlling access and permissions in the smart contracts, no one other than the data provider will have access to these records.

The address given to the applicant is unique and can only be used by the applicant. For creating the Ethereum smart contracts, Solidity ^0.5.0 [58] programming language is used. For

compiling the smart contracts, Truffle Suite [61] is utilized, and for migrating smart contracts for development on a local blockchain and further evaluation and tests, Ganache [19] was used and the MetaMask is connected to this blockchain. The connection between UI and blockchain is handled by an Ethereum JavaScript API named web3.js [62]. Figure 4 shows the overall connection and relations in the deployment and test phases.

For registration in the DCIaaS, applicants can access the application and register with their unique address given by MetaMask or any other crypto wallet, and then they can see available datasets. When registering, they must choose the "*Applicant*" role. Data providers must register as "*Data Provider*." Moreover, data providers can offer further information about themselves. For example, agencies and organization can register their name and title. After registering, they can inform visitors on their website that they are using the proposed

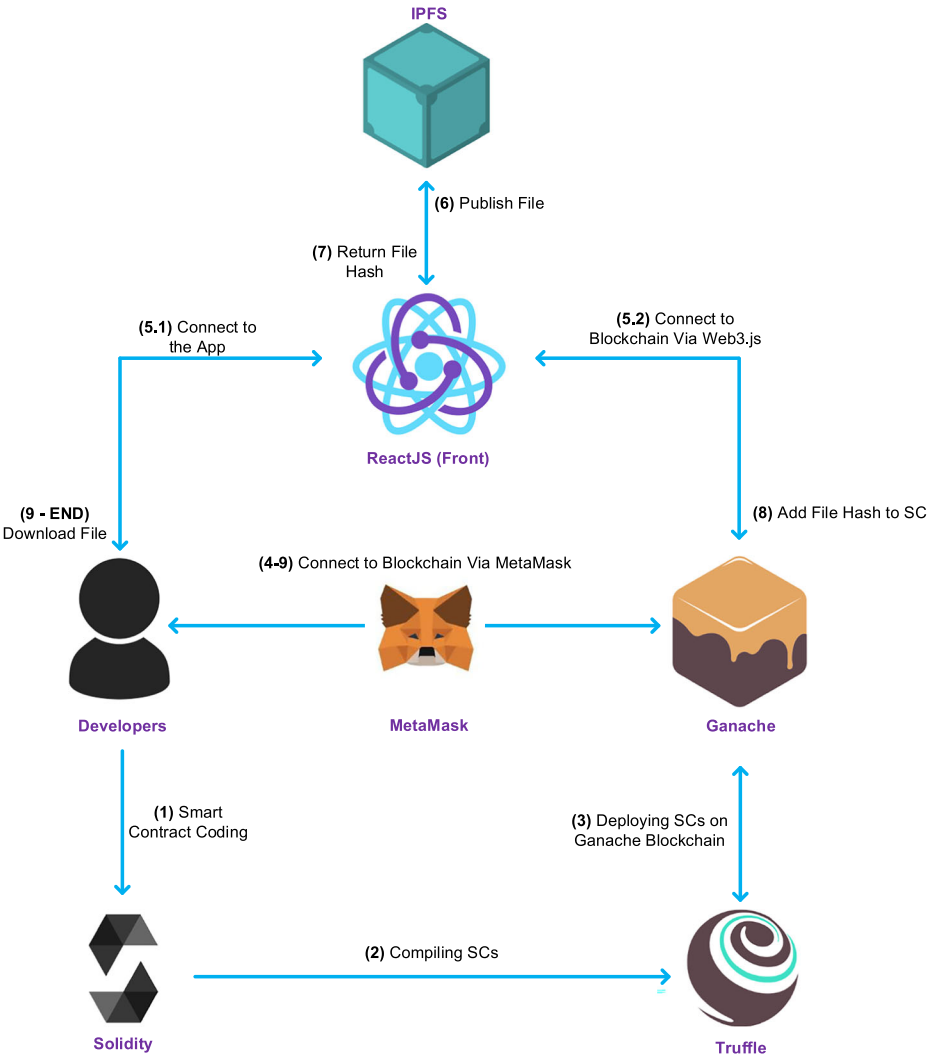


Fig. 4 Overall development connection

framework, and to avoid any possible abuse, they can share their address so that applicants can be sure whom they are going to work with.

As mentioned before, access control and permissions have been handled in the smart contract. Two functions (*accessPermited* and *accessRevoked*) are utilized to handle access and permissions. These functions utilize the implicitly available *msg.sender* from global variable *msg*, which contains the address of the *applicant/data provider* sending the transaction. By holding this address, the IPFS hash will only be available for the person who uploaded it (*Data Provider*) and the person who requested it (*Applicant*).

After the smart contract was tested and its functionality evaluated, it was deployed on an online test blockchain named Kovan Testnet Network [35]. Moreover, for uploading files on IPFS and later downloading them, an IPFS API named Infura [26] was utilized. Figure 5 shows the web application connected to the smart contract and blockchain with two datasets added to it. By clicking on "Request," the applicant will be redirected to a new page in which the applicant must provide a proposal of what he wants to do with the datasets and fill other requirements.

The final issue which should be considered in the implementation of the DCIaaS is the security of training the machine learning models. The security of the training process of machine learning models can be compromised as there are methods that determine whether an entity was used in the training set. For example, adversarial attacks called Member Inference, and Model Inversion [66] can reconstruct raw input data given the model's output. In theory, reconstructing a standard neural network to exploit input data seems unrealistic. In practice, however, there is always some real-world context which can be used to trace back the model to the input data. Publicly available datasets can also be linked to the original private and sensitive data [32].

In order to solve this problem, different solutions exist for training machine learning models while taking the privacy of the input data in mind. An effective method is using the TensorFlow Privacy (TFP) [43]. In order to use TFP, compared to standard TensorFlow, no changes to the model architectures and training procedures are required. Instead, to train models that protect the privacy of the training data, the hyper-parameters relevant to privacy, such as optimizers, are changed. Using a TFP optimizer that clips gradients according to a

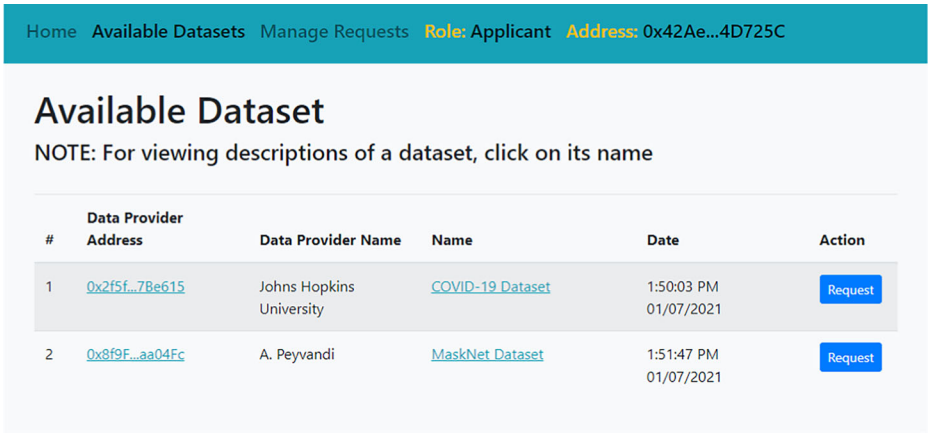


Fig. 5 The web application for applicant to view available datasets that were previously added to the smart contract

defined magnitude and adds noise of a defined size, the privacy of the training data can be protected. The TFP optimizers wrap the original TensorFlow optimizers. For example, when using Adam optimizer, TFP wraps it with its differential private counterpart (*DPAdamOptimizer*).

4 Experimental results

4.1 DCIaaS for lung cancer classification using histopathological images

In the first section of the experiments, the proposed framework is evaluated for medical applications. Since early diagnosis of cancer is crucial for treatment, as the first case study the proposed DCIaaS is used for lung cancer detection. The performance of the DCIaaS is compared with standard Stochastic Gradient Descent (SGD) method for training a CNN-based model on the lung cancer Histopathological images dataset [10]. For the training, 80% of the dataset was used, and the remaining 20% were selected for the test. One sample of some of the classes of the dataset is presented in Fig. 6.

In real-world scenarios of federated learning, each federated member (*Client*) will own its data locally. However, in this experiment, the entire dataset is stored in one place. In this simulation, five clients were considered, and therefore, the training data was randomly batched and split into five fragments, one for each client. For this case study, the EfficientNetB7 [60] neural network architecture, illustrated in Fig. 7, which is based on a weighted Bi-directional Feature Pyramid Network (BiFPN), was utilized for the classification task of the lung dataset. The top layers of the network were frozen, and the default initial weights were used. However, the output of the network was first fed to a *GlobalAveragePooling2D* layer followed by a *Flatten* layer, two fully connected layers of 128 and 64 neurons respectively, each activated by the ReLU function, and finally, an fully connected layer activated by *Softmax* acting as the final output of the architecture, classifies the dataset.

For training the models, SGD optimizer with a learning rate of 0.01 and accuracy as the metric was utilized. The global model was trained for 10 GI (epochs), meaning that the FedAvg algorithm (Fig. 2) is executed 10 times. The variable called w_0 is used to hold the initial weights, which comes from the weights of the global model. Next, a list of clients with correspondent addresses and data fragments of each participant is stored in a list called S_t. Then the first LI is executed. A local model Keras object for the current client is created and compiled, and the local model's weights are set to that of global weights of the ongoing GI. In

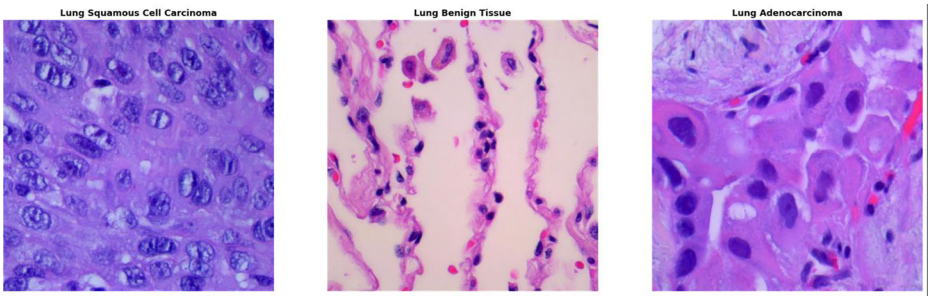


Fig. 6 Samples of the lung cancer Histopathological images dataset

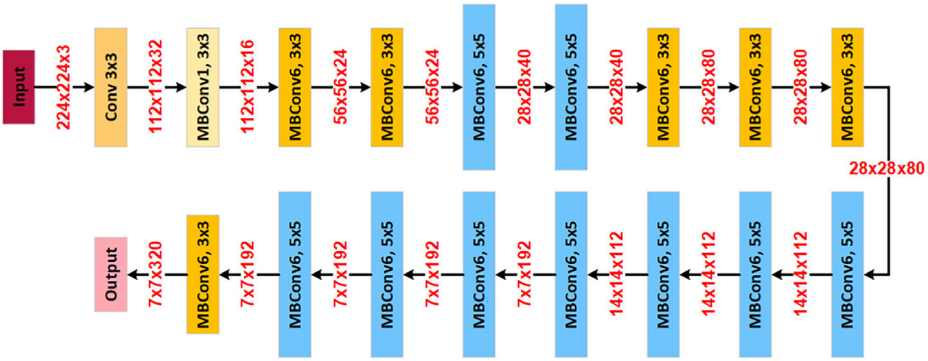


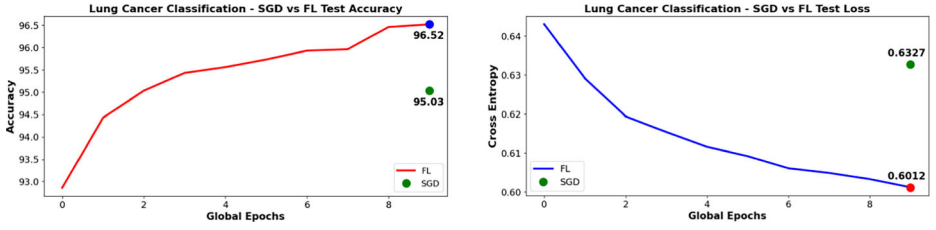
Fig. 7 EfficientNetB7 architecture

the next step, the local model will train only for one epoch on the client's data, and the acquired weights are added to a list, and the LI will be complete. For each GI, the LI is executed five times since five participants were considered for this simulation, and each client trains the model for one epoch per local iteration. After the clients trained their models locally for the current GI, it sums up all the weights acquired from the clients, takes the average, and sets the global model's weights to this average. Finally, the performance of the current model will be tested, and the current GI will be completed.

The performance of the federated learning-trained models is compared with a standard SGD trained model. As previously mentioned, the federated learning models were trained for 10 global epochs in total. Overall, the DCIaaS trained model for the classification of lung cancer showed better performance compared to that of the SGD trained model, which was also trained for 10 epochs. The DCIaaS trained model offered an accuracy of 96.52% on the test set, while the SGD-trained model has 95.0.% accuracy. Furthermore, the loss of the DCIaaS based trained model was 0.6012, which is lower than that of the SGD model at 0.6327. The comparison of the performance of these models trained for lung cancer classification is presented in Fig. 8.

Overall, the results suggest that the DCIaaS leads to better accuracy in training this model. In order to present the advantages and contributions of this research, the characteristics of the proposed framework are compared with recently proposed methods in Table 2.

In some of these researches, the aggregation of the trained models takes place on a centralized server, which makes results vulnerable to malicious activities. In other research, the sharing of trained models and weights does not take place on a secure channel. The



(a) Accuracy (b) Loss

Fig. 8 Performance comparison of SGD vs DCIaaS trained models for lung cancer classification. **a** Accuracy, **b** Loss

Table 2 Comparison with existing works

Reference	Blockchain	Federated Learning	IPFS	Access Control	Decentralized
DCIaaS	✓	✓	✓	✓	✓
Tsung-Ting et al. [37]	✓	×	×	✓	✓
Sheller et al. [55]	×	✓	×	✓	×
Dias et al. [15]	✓	×	×	✓	×
Zhang et al. [68]	×	✓	×	×	×
Naz et al. [47]	✓	×	✓	✓	✓
Rajendran et al. [53]	×	✓	×	×	✓
Liu et al. [39]	×	✓	×	×	×
Hasan et al. [24]	✓	×	✓	×	×

DCIaaS framework does not have the vulnerabilities of a federated learning-based method that may expose sensitive and personal data during their training process. Other researches that aim to eliminate problems of federated learning by utilizing blockchain technology are built around distributed systems, and the training data is still shared between miners or other individuals. This distribution of models and data can prove to be harmful to privacy. The proposed DCIaaS framework solves this issue as data holders are not required to share their data, models are trained on the data owner's end of the blockchain, and only the trained weights are shared in a secure manner.

Modelling and management of the smart environments in Society 5.0 require a vast amount of data [40]. The proposed DCIaaS framework has many applications for the participation of the data owners in training machine learning models for smart environments. As currently, one of the significant issues faced by the communities around the world is the COVID-19 pandemic; in this research, the application of the DCIaaS framework for training models required for management of discarded face masks which proved to be a major environmental and health hazard faced by governments during the pandemic is investigated. Previously, the applications of DCIaaS for computer aided diagnosis in COVID-19 pandemic is also investigated [50].

4.2 DCIaaS for smart city management in pandemic conditions

Training accurate machine learning models for smart city management in pandemic conditions requires the rapid and large-scale collection of data. The issue of privacy reduces the willingness of the crowd to share their data with academia and governmental agencies. In this case study, the application of DCIaaS for autonomous visual detection of littered face masks, which can act as an agent for the spread of the virus, is demonstrated.

Litter management is one of the significant tasks that should be addressed in smart environments. The discarded face masks can lead to the possible spread of the virus through intermediary agents. In order to investigate this problem, we collected a new dataset called MaskNet (https://github.com/Tenebris97/MaskNet), in Austria and Iran during July 2020. The dataset was collected daily for seven days in cities like Steyr, Linz, Wels, and Tehran, during different times of day from 6 A.M. up to 6 P.M. from different environments such as streets, parks, riverbanks, inside buildings, and offices. The dataset consists of 1058 surgical mask images that are littered on the streets and other urban areas. We assume that there is a researcher (applicant) who wants to use this MaskNet dataset to train an object detection

model which can detect masks in different environments. We assume that due to legal and privacy concerns, the dataset cannot be shared online. However, the applicant can use the proposed DCIaaS framework to train the required deep learning model without being concerned about legal and privacy issues. Using DCIaaS, the smart city management can train various required machine learning models on datasets created by citizens. Figure 9 demonstrates this scenario.

In this scenario, we play the role of the data provider, and an applicant wants to train an object detection model using the MaskNet dataset. After the applicant sends his request and we accept it, he uploads his code on IPFS, and the hash will be stored on the smart contract after it is validated (by the smart contract). Then, we (the data provider) will receive the hash, download the file, and train the object detection model. After the training process is finished, we upload the final checkpoint and the model itself (protobuf or *pb* file, for example) on IPFS. Moreover, the rest of the necessary files, such as the pipeline config file, will be uploaded by us on IPFS, and their respective hash will be shared via the smart contract. Then, the applicant will receive the hashes and download the files. Now, the applicant can test the object detection model on any images that are not in dataset, as shown in Fig. 10.

The experiments in this scenario are performed using the previously-discussed EfficientNetB7 architecture. For experimental results, the architecture was retrained

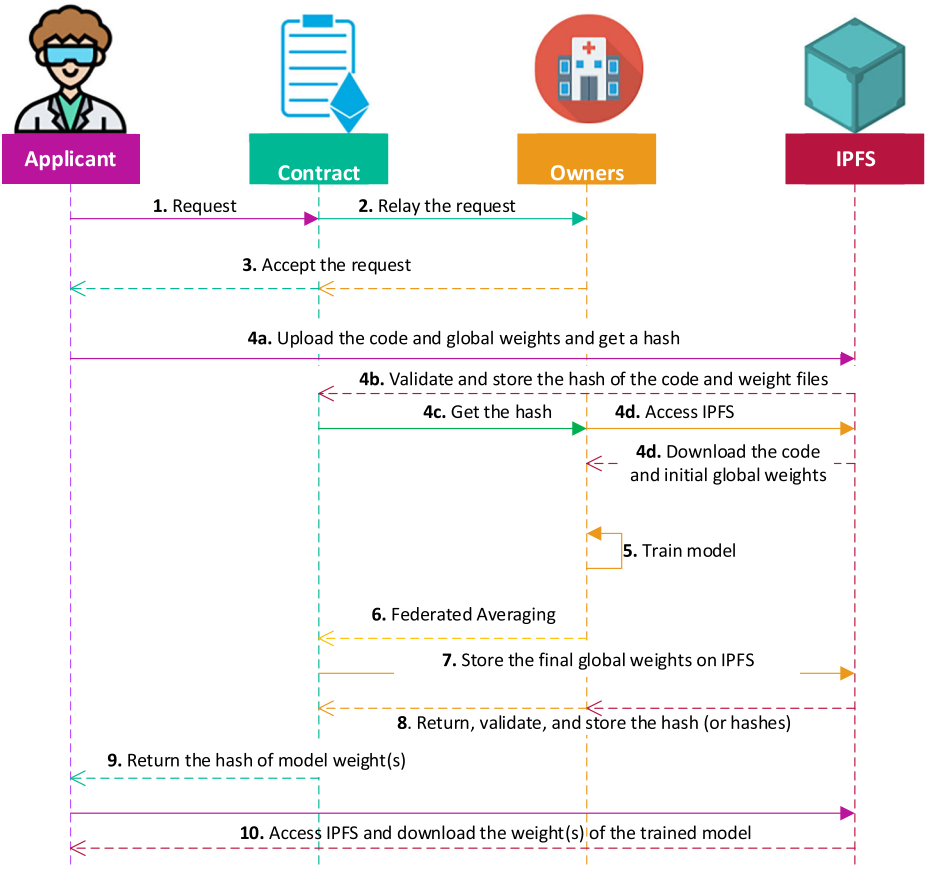
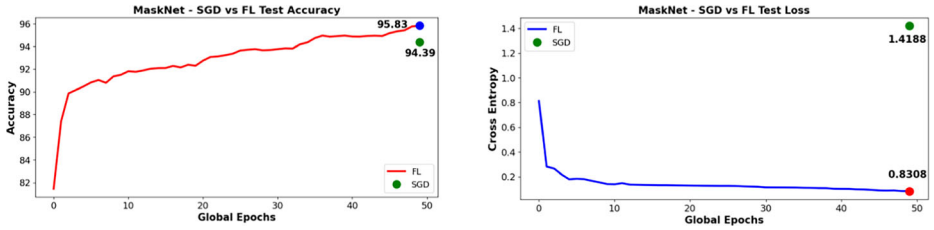


Fig. 9 The sequence diagram of the proposed DCIaaS framework

using TensorFlow Object Detection API for 18,000 steps. In order to train a model on the MaskNet dataset, both using the vanilla SGD and DCIaaS based federated learning, transfer learning using the EfficientNetB7 architecture was utilized. This architecture, with approximately 66 million parameters, has presented finer results compared to other widely used architectures in the literature such as Inception, Xception, DenseNet, NASNet, and ResNet, when trained on the ImageNet. By utilizing the proposed DCIaaS framework, a federated learning model was trained on the MaskNet dataset, and its performance was compared with a standard SGD model. The DCIaaS based model was trained for 50 global epochs, 10 per client, and it offered a better performance compared to that of the SGD-trained model, also trained for 50 epochs. For the SGD trained model, the training accuracy of 94.39% was achieved. The DCIaaS trained model on the MaskNet dataset offered an accuracy of 95.83% on the test set. Moreover, the loss of the DCIaaS trained model (0.0831) was notably lower than that of the SGD trained model (1.4188). The test accuracy and test loss of the DCIaaS and SGD trained models on the MaskNet dataset are presented in Fig. 11.

It should be considered that in real-life and more complex scenarios, these contrast between the results can be higher as federated data held by distributed clients can keep samples of minority classes which allows the final data used for training the machine learning models to contain enough samples from various categories and specific cases. Another issue that should be considered is implementation of DCIaaS on the IoT. Software Defined Networks (SDN) [49] can be utilized for management of data collection and performing distributed and decentralized machine learning.

Fig. 10 Testing the face mask detection model on Google Images



(a) Accuracy (b) Loss

Fig. 11 Performance comparison of SGD vs DCIaaS models trained on the MaskNet dataset. **a** Accuracy, **b** Loss

4.3 Formal verification of the efficiency of DCIaaS

As the proposed framework is a software framework, the analysis of the applicability and performance of the DCIaaS is in the domain of software performance evaluation and verification of software dependability. As the proposed framework is a combination of software entities such as smart contracts on the blockchain, the formal verification of these software components can prove the dependability of the DCIaaS framework. Even though blockchain provides a secure and trusted environment for executing and storing smart contracts, these contracts are still vulnerable, and possible attacks and bugs might exploit them. Therefore, in order to solve potential issues in smart contracts, they should be verified before being deployed on a blockchain network. This verification ensures that the smart contracts will be executed according to the intended parameters.

The formal verification of smart contracts is investigated and proven using various methods. Bai et al. [5] introduced a formal modelling and verification method to verify the properties of smart contracts using SPIN, a model checker tool. By utilizing this tool, the authors were able to verify the correctness and necessary properties of smart contracts. For the Ethereum based smart contracts similar to the smart contracts used in DCIaaS, Yang and Lei [64] proposed a formal symbolic process for verifying the reliability and security of Ethereum smart contracts using a formal proof management tool named Coq proof assistant. Abdellatif et al. [2] proposed formal modelling and verification of smart contracts based on the users' behaviour on a blockchain network, which verifies a smart contract's behaviours in its execution environment. Beillahi et al. [7] proposed an automated method for verifying smart contracts based on the functional properties of a smart contract. Even a standard runtime verification approach can be utilized to support the dependability and correctness of smart contracts during their runtime.

In the Solidity verification tool, which is used in DCIaaS, a Satisfiability Modulo Theories (SMT) based formal verification module is used for verification of the smart contracts [4]. This verification tool is integrated into the Solidity compiler, and during compilation, warns for potential fails. Similarly, Solv-verify [23] is a source-level verification tool that is built on top of the Solidity compiler and automates Solidity smart contracts' verification based on SMT solvers. For the general verifiability of the smart contracts, by utilizing the Markov decision process (MDP) and game theory, it is proven that smart contracts are verifiable [8].

5 Conclusions and future work

In this paper, a blockchain-based federated learning framework for privacy-preserving machine learning, called DCIaaS, is proposed. In the proposed decentralized blockchain-based framework, the models will be trained on the data provider's side and off-chain using federated learning, and only the learned model parameters and weights will be shared on the blockchain and using the smart contracts. Experimental results show an increase in accuracy of the models trained using the DCIaaS framework compared to decentralized training. As a case study, the DCIaaS framework is utilized for medical and smart city applications related to Society 5.0. For the future work of the proposed framework, decentralized agent-based modelling will be implemented. Current simulation models for autonomous vehicles, drones, and robots extensively rely on centralized models. However, such an approach can target security and privacy. A blockchain-based and agent-based simulator for smart cities, which considers the communication between agents through smart contracts, can address this issue. This decentralized agent-based model can use privacy-preserved data to model complex scenarios more accurately. The further expansion of DCIaaS can include agent-based modelling using a decentralized blockchain network.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Abbasi MH et al (2019) Deep visual privacy preserving for internet of robotic things. In: 2019 5th conference on knowledge based engineering and innovation (KBEI). IEEE
2. Abdellatif T, Brousmiche K-L (2018) Formal verification of smart contracts based on users and blockchain behaviors models. In: 2018 9th IFIP international conference on new technologies, mobility and security (NTMS). IEEE
3. Aledhari M, Razzak R, Parizi RM, Saeed F (2020) Federated learning: a survey on enabling technologies, protocols, and applications. IEEE Access 8:140699–140725
4. Alt L, Reitwießner C (2018) SMT-based verification of solidity smart contracts. In: International symposium on leveraging applications of formal methods. Springer
5. Bai X et al 2018 Formal modeling and verification of smart contracts. In: Proceedings of the 2018 7th International Conference on Software and Computer Applications
6. Baldominos A, Saez Y (2019) Coin. AI: A proof-of-useful-work scheme for blockchain-based distributed deep learning. Entropy 21(8):723
7. Beillahi SM et al (2020) Behavioral simulation for smart contracts. In: Proceedings of the 41st ACM SIGPLAN Conference on Programming Language Design and Implementation
8. Bigi G et al (2015) Validation of decentralised smart contracts through game theory and formal methods. In: Programming languages with applications to biology and security. Springer, pp 142–161
9. Blanchard P et al (2017) Machine learning with adversaries: Byzantine tolerant gradient descent. In: Proceedings of the 31st International Conference on Neural Information Processing Systems
10. Borkowski AA et al (2019) Lung and colon cancer histopathological image dataset (lc25000). arXiv preprint arXiv:1912.12142
11. Brakerski Z, Gentry C, Vaikuntanathan V (2014) (Leveled) fully homomorphic encryption without bootstrapping. ACM Trans Comput Theory 6(3):1–36
12. Briggs C, Fan Z, Andras P (2021) A review of privacy-preserving federated learning for the internet-of-things. Federated Learning Systems, p 21–50
13. Brisimi TS, Chen R, Mela T, Olshevsky A, Paschalidis IC, Shi W (2018) Federated learning of predictive models from federated electronic health records. Int J Med Inform 112:59–67

14. Cabrero-Holgueras J, Pastrana S (2021) SoK: privacy-preserving computation techniques for deep learning. Proc Priv Enh Technol 4:139–162
15. Dias JP et al (2018) Blockchain for access control in e-health scenarios. arXiv preprint arXiv:1805.12267
16. Dwork C, Roth A (2014) The algorithmic foundations of differential privacy. Found Trends Theor Comput Sci 9(3–4):211–407
17. Entriken W. Introduction to smart contracts. [cited 2020 November 30]; Available from: https://ethereum.org/en/developers/docs/smart-contracts/.
18. Fadaeddini A, Majidi B, Eshghi M (2020) Secure decentralized peer-to-peer training of deep neural networks based on distributed ledger technology. J Supercomput 76:10354–10368
19. *Ganache*. [cited 2020 December 17]; Available from: https://www.trufflesuite.com/ganache.
20. Ge S et al (2020) Fedner: Privacy-preserving medical named entity recognition with federated learning. arXiv preprint arXiv:2003.09288
21. Gilad Y et al (2017) Algorand: scaling byzantine agreements for cryptocurrencies. In: Proceedings of the 26th symposium on operating systems principles
22. Goel A et al (2019) DeepRing: protecting deep neural network with blockchain. In: Proceedings of the IEEE conference on computer vision and pattern recognition workshops
23. Hajdu Á, Jovanović D (2019) Solc-verify: a modular verifier for solidity smart contracts. In: Working conference on verified software: theories, tools, and experiments. Springer
24. Hasan HR, Salah K (2018) Proof of delivery of digital assets using blockchain and smart contracts. IEEE Access 6:65439–65448
25. Hitaj B, Ateniese G, Perez-Cruz F (2017) Deep models under the GAN: information leakage from collaborative deep learning. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security
26. *Infura*. [cited 2020 December 15]; Available from: https://infura.io/.
27. Kim JMJ (2014) Stellar. Available from: https://www.stellar.org/
28. Kaissis GA, Makowski MR, Rückert D, Braren RF (2020) Secure, privacy-preserving and federated machine learning in medical imaging. Nat Mach Intell 2(6):305–311
29. Karki D. Can you guess how much data is generated every day? [cited 2021 Januray 7]; Available from: https://www.takeo.ai/can-you-guess-how-much-data-is-generated-every-day/
30. Kasiviswanathan SP, Smith A (2014) On the'semantics' of differential privacy: a bayesian formulation. J Priv Confid **6**(1)
31. Keccak. Keccak Team. [cited 2021 August 28th]; Available from: https://keccak.team/keccak.html.
32. Keydana. RStudio AI Blog: Hacking deep learning: model inversion attack by example. [cited 2020 December 14]; Available from: https://blogs.rstudio.com/tensorflow/posts/2020-05-15-model-inversion-attacks/.
33. Khan LU, Saad W, Han Z, Hossain E, Hong CS (2021) Federated learning for internet of things: recent advances, taxonomy, and open challenges. IEEE Commun Surv Tutor 23(3):1759–1799
34. Konečný J et al (2016) Federated optimization: distributed machine learning for on-device intelligence. arXiv preprint arXiv:1610.02527
35. *Kovan Testnet*. [cited 2020 December 15]; Available from: https://kovan-testnet.github.io/website/.
36. Kumar R et al (2021) Blockchain based privacy-preserved federated learning for medical images: a case study of COVID-19 CT scans
37. Kuo T-T, Ohno-Machado L (2018) Modelchain: decentralized privacy-preserving healthcare predictive modeling framework on private blockchain networks. arXiv preprint arXiv:1802.01746
38. Li T, Sahu AK, Talwalkar A, Smith V (2020) Federated learning: challenges, methods, and future directions. IEEE Signal Process Mag 37(3):50–60
39. Liu B et al (2020) Experiments of federated learning for covid-19 chest x-ray images. arXiv preprint arXiv:2007.05592
40. Majidi B et al (2021) Geo-spatiotemporal intelligence for smart agricultural and environmental eco-cyber-physical systems. In: Enabling AI applications in data science. Springer, pp 471–491
41. Mallaki M, Majidi B, Peyvandi A, Movaghar A (2021) Off-chain management and state-tracking of smart programs on blockchain for secure and efficient decentralized computation. Int J Comput Appl:1–8
42. McMahan B et al (2017) Communication-efficient learning of deep networks from decentralized data. In artificial intelligence and statistics. PMLR
43. McMahan HB et al (2018) A general approach to adding differential privacy to iterative training procedures. arXiv preprint arXiv:1812.06210
44. Melis L et al (2018) Inference attacks against collaborative learning. arXiv preprint arXiv:1805.04049 13
45. Menezes AJ, Van Oorschot PC, Vanstone SA (2018) Handbook of applied cryptography. CRC press
46. *MetaMask*. [cited 2020 December 17]; Available from: https://metamask.io/.

47. Naz M, al-zahrani FA, Khalid R, Javaid N, Qamar AM, Afzal MK, Shafiq M (2019) A secure data sharing platform using blockchain and interplanetary file system. Sustainability 11(24):7054
48. Nguyen HT, Sehwag V, Hosseinalipour S, Brinton CG, Chiang M, Vincent Poor H (2021) Fast-convergent federated learning. IEEE J Sel Areas Commun 39(1):201–218
49. Norouzi A, Majidi B, Movaghar A (2018) Reliable and energy-efficient routing for green software defined networking. In: 2018 9th international symposium on telecommunications (IST). IEEE
50. Peyvandi A et al (2021) Computer-aided-diagnosis as a service on decentralized medical cloud for efficient and rapid emergency response intelligence. N Gener Comput:1–24
51. Peyvandi A, Majidi B, Peyvandi S (2022) Blockchain-based secure biomedical data-as-a-service for effective internet of health things enabled epidemic management. In: Kose U et al (eds) Computational intelligence for covid-19 and future pandemics: emerging applications and strategies. Springer Singapore, Singapore, pp 405–424
52. Phong LT, Aono Y, Hayashi T, Wang L, Moriai S (2018) Privacy-preserving deep learning via additively homomorphic encryption. IEEE Trans Inf Forensics Secur 13(5):1333–1345
53. Rajendran S, Obeid JS, Binol H, D Agostino R Jr, Foley K, Zhang W, Austin P, Brakefield J, Gurcan MN, Topaloglu U (2021) Cloud-based federated learning implementation across medical centers. JCO Clin Cancer Inform 5:1–11
54. Shayan M et al (2018) Biscotti: A ledger for private and secure peer-to-peer machine learning. arXiv preprint arXiv:1811.09904
55. Sheller MJ, Edwards B, Reina GA, Martin J, Pati S, Kotrotsou A, Milchenko M, Xu W, Marcus D, Colen RR, Bakas S (2020) Federated learning in medicine: facilitating multi-institutional collaborations without sharing patient data. Sci Rep 10(1):1–12
56. Shokri R, Shmatikov V (2015) Privacy-preserving deep learning. In: Proceedings of the 22nd ACM SIGSAC conference on computer and communications security
57. Shokri R et al (2017) Membership inference attacks against machine learning models. In: 2017 IEEE symposium on security and privacy (SP). IEEE
58. *Solidity*. [cited 2020 December 17]; Available from: https://docs.soliditylang.org/en/v0.5.0/resources.html.
59. Song C, Ristenpart T, Shmatikov V (2017) Machine learning models that remember too much. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security
60. Tan M, Le Q (2019) Efficientnet: rethinking model scaling for convolutional neural networks. In: International conference on machine learning. PMLR
61. *Truffle Suite*. [cited 2020 December 17]; Available from: https://www.trufflesuite.com/.
62. *web3.js*. [cited 2020 December 17]; Available from: https://web3js.readthedocs.io/en/v1.3.0/.
63. Weng J et al (2019) Deepchain: Auditable and privacy-preserving deep learning with blockchain-based incentive. IEEE Trans Dependable Secure Comput
64. Yang Z, Lei H (2018) Formal process virtual machine for smart contracts verification. arXiv preprint arXiv:1805.00808
65. Zhang Y, Van der Schaar M (2012) Reputation-based incentive protocols in crowdsourcing applications. In: 2012 proceedings IEEE INFOCOM. IEEE
66. Zhang Y et al (2020) The secret revealer: generative model-inversion attacks against deep neural networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition
67. Zhang C, Xie Y, Bai H, Yu B, Li W, Gao Y (2021) A survey on federated learning. Knowl-Based Syst 216:106775
68. Zhang W, Zhou T, Lu Q, Wang X, Zhu C, Sun H, Wang Z, Lo SK, Wang FY (2021) Dynamic Fusion-based Federated Learning for COVID-19 Detection. IEEE Internet Things J:1
69. Zhao Y, Zhao J, Jiang L, Tan R, Niyato D, Li Z, Lyu L, Liu Y (2020) Privacy-preserving blockchain-based federated learning for IoT devices. IEEE Internet Things J 8(3):1817–1829
70. Zhou J et al (2021) A survey on federated learning and its applications for accelerating industrial internet of things. arXiv preprint arXiv:2104.10501
71. Zhu T et al (2017) Preliminary of differential privacy. In: Differential privacy and applications. Springer, pp 7–16