Decentralized Federated Learning on a High-performance Computing Platform: Blockchain-based Security and Optimization

Yejin Kwon[1], Jeongcheol Lee[1] and Yongbom Park[2,*]

[1]*Department of Supercomputing Acceleration Research, Division of National Supercomputing R&D, Korea Institute of Science and Technology Information, South Korea*
[2]*Department of Software Engineering, Dankook University, South Korea*
E-mail: yejinkwon@kisti.re.kr; jclee@kisti.re.kr; ybpark@dankook.ac.kr
[*]*Corresponding Author*

Received 17 July 2025; Accepted 28 August 2025

Abstract

As various artificial intelligence (AI)-related services emerge, attempts to apply AI models to industrial and service fields are ongoing. In preparation for the increase in demand for such AI services, the importance of data required to create AI models is increasing. As data for applying services is a key element in training AI models and improving performance, the privacy and security of data have also recently emerged as significant issues. Although various AI learning platforms exist, users must share data that they have personally created and refined. Consequently, users who do not want to share personal data tend to be reluctant to access and use these platforms. Therefore, in this paper, we built a platform that can create and share specific AI models without sharing data. We designed and built a platform that can perform blockchain-based federated learning to ensure the authenticity and privacy of the user's learning model and global model,

Journal of Web Engineering, Vol. 24_8, 1231–1262.
doi: 10.13052/jwe1540-9589.2483

allowing the learning of the final result model according to the sharing model of each dataset during each learning session. Each user learns based on their personal dataset, and the weights are integrated based on the learned models to create a single global model. The contribution of each user is measured based on the blockchain-based model creation, leading to the development of a high-performance AI model. We conducted experiments using the MNIST and COVID-19 datasets, focusing on data independence. The results showed that significant results could be achieved with just 10–20% data sharing. These experimental results confirmed that federated learning is possible in an environment where data independence is maintained and individual users do not share data.

By applying this learning method to the platform, it is expected that the collaboration barriers between researchers in computational science and engineering will be lowered and there will be an increase in the usability of integrated research.

Keywords: HPC, federated learning, blockchain, simulation.

1 Introduction

Scientists consider computational simulation data as important and private research data. There have been many attempts to use simulation data in computational science on high-performance computing (HPC) clusters, but scientists are reluctant to share their simulation data because the platform or cloud service cannot guarantee the privacy and confidentiality of their data. As a result, research in conjunction with the differences in applied fields has not improved simulation services with HPC cluster platforms. The scalability of a service platform that produces HPC simulation data with artificial intelligence (AI) [3] is low, and scientists are hesitant to share their data. In particular, because the service platform for AI requires data sharing during model learning, scientists who regard the privacy and confidentiality of their data as very important tend to avoid such platforms. The platform for AI services provides a learning method with encrypted data [4] or an interchange model without a dataset for the district environment [1, 2]. In contrast, if the AI model learns with convergence and various datasets, it can achieve high performance and utilize multiple specialties. It is essential for the AI model to learn from a large number of private simulation datasets across various categories [5]. The requirements of the scientists and the service platform are contrary, so this paper suggests an HPC cluster platform that guarantees

private simulation data. We provide a separate service with the learning model and dataset when training the AI model with various applied field data. Each user can submit a batch job on an HPC platform for computational science simulation or for training an AI model with privately protected simulation data. The AI model with simulation data is registered in a private blockchain network, allowing the user to find and share the registered model on the blockchain. This dataset is not shared on the blockchain network. The initial global model, where all weights are initialized to 0, along with the evaluation dataset and test dataset, are shared on the blockchain network through our platform. When the registered model on the blockchain is used, the user can confirm the resistance model because the verification result of the AI model based on each user's procedural simulation data is verified.

A shared dataset and shared model for a specific model must be provided to learn from the data without sharing private data based on personally generated data. The definition of such a shared model can be stored and shared by the project unit for the same problem or by using the same model. Information about the shared model and shared dataset can also be stored on the blockchain. Due to the continuous model updates, each shared model can undergo an evaluation process based on the federated learning results of the models learned locally, update the shared model, and redistribute the updated model to users who use the same model for further learning.

To create an AI model without sharing personally created experimental data, the model is trained locally based on the user's private data, evaluated, and the results that show a certain level of performance or higher are stored on the blockchain. In addition, a federated model was created that combines learning models trained by multiple users into a single model and shares it.

In this study, we designed a blockchain-based federated learning process to share and learn AI models based on an HPC platform and built global learning, local learning, and blockchain-based federated learning models for models shared on the platform. Existing federated learning approaches have focused on real-time learning using various sensor data or integrating multiple resources to perform learning. In this study, we designed and built a platform capable of performing federated learning in an environment where data is not shared between users while maintaining data independence. Unlike existing learning platforms, we designed blockchain-based model learning and sharing to verify the learning performance of the learning model through the platform. We implemented a method to verify and learn the generated AI model by registering the learning model on the blockchain and adjusting the learning model of the global model according to the shared contribution of

the learning model registered on the blockchain and the learning contribution of the entire blockchain.

The structure of this paper is as follows.

- Section 2 reviews the related work, focusing on deep learning based on blockchain. Section 3 presents the algorithms for federated learning using individual users' data and describes the approach of collaborative learning with mutually independent datasets.
- Section 4 provides the detailed design and algorithms of federated learning on the HPC-based web platform.
- Section 5 reports the experimental results obtained from simulating federated learning on the HPC platform's resources. Finally, the conclusion summarizes the overall study and discusses directions for future research.

2 Related Work

2.1 Research on Utilizing Deep Learning Based on Blockchain

In this study, we provide a platform that curates the simulation result data generated from the HPC simulation-based platform and stores and shares the models generated based on the data. The currently constructed HPC platform-based simulation system is designed so that each user can execute HPC simulations and curate the executed result data, ensuring that the simulation result data preserves the individuality of the simulation result data in each field of computational science and engineering. The simulation results are very sensitive to individual authentication and sharing methods [37]. In addition, although we recognize the need for data sharing, we are interested in guaranteeing the originality and authenticity of individually generated simulation data and the security of individual data. In general, the simulation system provided on the HPC-based simulation platform provides individual simulation result data curation or an additional calculation process for the result data but does not include the process for the ownership of the result data and the data authenticity for data sharing. Therefore, to add a platform process for the storage and distribution of models for the simulation result data without various authentications and complex procedures for the result data from which the simulation was executed, we added a blockchain-based model storage and sharing process and designed and constructed a data storage and sharing method for the simulation result data based on the existing web-based data process.

The direction of research on applying blockchain to existing AI generation models has been focused on supplementing research concerning on the falsification of AI model weights, which has recently become an issue, as well as the dispute over ownership of the user who created the model. It is used as a method to prove the integrity of the data using blockchain, which guarantees authenticity and proves that it has not been falsified, and to prove the ownership of the model and the authenticity of the developer. For example, research is being conducted on a framework that joins a blockchain network, creates an ID for each object tracked in the multi-object tracking framework, and tracks and manages the object using a smart contract [6]. For the multi-object tracking framework, an object tracking framework based on blockchain was proposed and built to register and manage objects, maintaining the reliability and tracking of unfalsified objects. In another study that combined AI and blockchain models, research was conducted on the development of a new blockchain technology protocol using deep reinforcement learning for blockchain mining and smart contracts. In that study, the processing capacity of the existing blockchain framework was increased by applying deep reinforcement learning to the transaction and voting protocol, block mining, and the process used for each blockchain protocol, and it was proven that the delay speed required to apply blockchain transactions can also be processed dramatically [8]. Similarly, a study using deep reinforcement learning was conducted to solve the power and resources required for blockchain mining by sharing edge-computing resources [7]. In this edge-computing method using smart devices, communication and resource utilization between different devices are proposed using blockchain and deep reinforcement learning, and a protocol for managing edge devices and utilizing resources based on devices registered in the blockchain is proposed. Research that combines blockchain and deep learning is continuously being conducted in various fields, and it can be seen that the convergence of these various technologies has supplemented the problems of existing deep learning research, secured the reliability of additional data itself, and increased efficiency by conducting procedures such as additional mutual authentication protocols through blockchain.

As the federated learning model, in which many users jointly create a specific AI model, has recently begun to receive attention, organic learning methods that integrate the resources of various users, edge devices, and endpoint devices to create a single model have also begun to receive attention. Studies have been conducted on frameworks that manage each device or user by integrating them with a blockchain to assist in the learning method

and ensure reliability [9]. In addition, federated learning methods utilizing blockchain are widely used, such as learning using edge devices or cloud computing resources, and linked federated learning methods can utilize the resources of each user who uses various resources and computational capabilities. Research is being conducted as a way to secure reliability and integrity by managing arbitrary registration and withdrawal of edge devices or users represented by each computational resource using blockchain [10, 12].

Therefore, in this study, we designed and developed a framework that provides federated learning functions based on blockchain to ensure the confidentiality and integrity of simulation data in the fields of computational science and engineering. We built and tested a platform that uses blockchain techniques to register and share models generated from federated learning results and implement smart contracts. The platform provides a framework for web-based HPC-based simulations and various preprocessors/postprocessors and service functions for data analysis for the convenience of users. Currently, as a condition for running a simulation, various services are provided that allow users who have logged in as registered users to run simulation applications registered on the current platform and check the execution results. When a simulation is run, a model is generated based on a specific AI model as a service on the platform to support federated learning. Subsequently, the corresponding model can be verified and evaluated based on the blockchain. Because users who run each simulation can use only the weights of the model to predict data without sharing information about the data they personally generate, privacy is guaranteed for the data that each user does not want to share. Federated learning can be performed by registering only specific AI models in the blockchain. This process is illustrated in Figure 1.

Based on the individually generated data, simulations were run through batch jobs on the HPC cluster, AI models were trained, and a model was created. The generated learning model is registered in an IPFS-based database. Because it is difficult to store large amounts of data in a blockchain transaction, the AI model is shared using IPFS, which stores large amounts of data and allows access to the data with only the simple content ID of the stored AI model. Once the AI model is registered, individual users of the platform can register it as a blockchain-based AI model, and the registration process is registered as a transaction within the blockchain network. In addition, when various users accumulate learning models for the same AI model until they reach a certain threshold, federated learning is executed based on the accumulated AI model. Federated learning is performed based

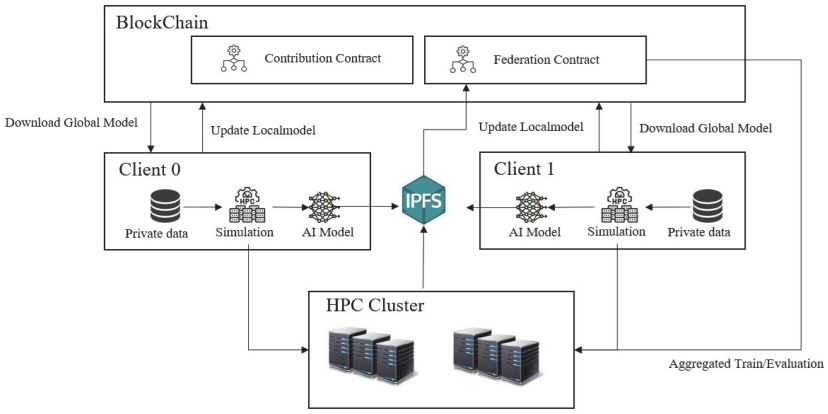


Figure 1 Federated learning architecture on an HPC platform.

on the transactions in the stored block, the federated learning result of the AI model is evaluated, and the federation contract process is performed. This process is also performed as a batch job based on HPC, and registration is performed through the evaluation process of the client, which creates an AI model based on the execution results.

Among the methods for AI model learning, the learning method using federated learning is widely used in the form of learning based on various local sensor devices or local learning resources, transmitting local data to a central server, or organically integrating local and central cluster resources. Blockchain is the foundational technology used to maintain this decentralized learning method [9, 10]. The platform is built by registering local sensor devices or edge devices, which are represented by various clients, to the blockchain and learning from the data updated based on the devices registered in the blockchain. In addition, blockchain is used as the basic framework that continuously transmits data for AI model learning based on various devices and guarantees reliability and integrity for data sharing. The detailed algorithm and techniques used for federated learning are described in the following sections.

3 Federated Learning

3.1 Algorithm for Federated Learning

Federated learning comprises various AI learning structures. Federated learning, which has been studied extensively recently, uses the resources of various

devices, such as various sensors, edge, and endpoint devices in cloud computing, to conduct AI learning by combining the resources of multiple devices [11, 12]. In addition, in the case of various sensor networks, there is a method of integrating the collected data and learning an AI model by conducting real-time analysis on sensor data transmitted locally. In this data integration method, there is a method of transmitting and integrating the gradient learned locally to the server for each learning branch [2], a method of combining weights according to the distribution of data, a method of applying weights for each local model of each client, and a method of creating a global model that improves performance by adding a regularization term.

The learning methods for corresponding federated learning are summarized in Table 1.

- **FedAvg** is an algorithm that creates a global model by averaging the weights of the models calculated in each client. Although it is a simple algorithm, it has a higher accuracy than expected based on the structure of the same model and the characteristics of the shared data. Because the algorithm is simple and does not require much computation, it is widely used in federated learning. The FedAvg algorithm has been used as a basic algorithm in other advanced algorithms. As shown in Table 1,

Table 1 Algorithm list for federated learning

Algorithm	Definition	Learning
FedAvg	Each client trains a local model, and the server averages the updates to create a global model [13–16]	Localized learning
FedProx	An extension of FedAvg that introduces a proximal term in the local optimization function to mitigate the impact of heterogeneous data across clients [17–19]	Localized learning
FedSGD	Averages gradient updates from each client on the server [2, 20–24]	Centralized learning
FedMA	Constructs the shared global model in a layer-wise manner by matching and averaging hidden elements (i.e., channels for convolution layers; hidden states for LSTM; neurons for fully connected layers) with similar feature extraction signatures [27–30]	Localized and centralized learning
FedOpt	A variant of FedAvg that applies advanced optimization techniques (e.g., Adam, SGD+Momentum) on the server to improve convergence speed [25, 26]	Localized and centralized learning
FedHEAL	FedHEAL ultimately achieves performance fairness under domain skew [38]	Localized learning

algorithms such as FedProx and FedOpt have been proposed as algorithms that learn AI models by adding additional factors that expand or flatten the distribution of the entire dataset based on the existing FedAvg algorithm and show higher performance than the existing FedAvg.

- **FedProx** is an improved version of FedAvg and is an algorithm proposed to solve the problem of data imbalance between clients. It adds a regularization term (proximal term) that adjusts the deviation of the model and calculates it by including it in the loss function to prevent significant deviation from the global model.
- **FedSGD** updates the model based on stochastic gradient descent using a single batch size of data from each client and sends this SGD to the server to perform the update. It is an algorithm that learns data with a single batch size locally, sends the gradient calculated in the process to the server, integrates the gradients sent from clients on the server, calculates the average, and updates the global model. This method can be efficient in a client–server structure with limited resources for continuously producing data and performing AI learning, such as small endpoint devices or sensor devices that are continuously updated.
- **FedMA** calculates the average value in the hidden layer of each neural network for deep learning models, particularly in deep learning structures with neural network architectures such as LSTM or CNN, which can be adjusted in neural networks with different structures. In other words, in the deep learning architecture of the neural network structure, the weights are adjusted by matching the neurons of each layer so that the weights of the models can be combined more efficiently by adjusting the matching values in neural networks with different structures, which enables improved performance. However, this algorithm may require additional resources because computational complexity increases, and the amount of computation that needs to be learned increases.
- **FedOpt** applies an additional optimization function when merging the models learned locally on the server side. That is, the server applies the FedAvg algorithm to the weights of the clients and then applies the optimization function once more according to the data type while merging the weights. Depending on the optimization method used, they are classified as FedAdam [31, 32], FedYogi [32], or FedAdagrad [32].
- **FedHeal** short for Federated HEALing, is a federated learning method that mitigates parameter update conflicts and ensures fairness in model aggregation in environments with domain bias. It reduces damage to specific domains and maintains fair global performance [38, 39].

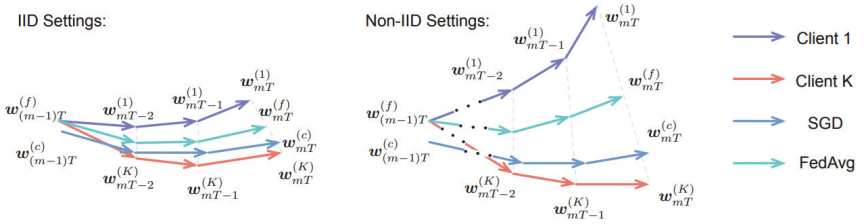


Figure 2 Illustration of weight divergence for federated learning with IID and non-IID data [33].

3.2 Non-IID (Independent and Identically Distributed) Dataset

The AI learning algorithm for integrating the data used by various clients who perform federated learning can be based on various algorithms, as described above. The various clients that require federated learning and the structures of the models and datasets used by the clients can vary for each client and depend on the type of server that leads the federated learning.

To create a single prediction model by combining models from various clients, there may be shared dataset structures and models among clients, and there is a possibility that there is no shared data structure among clients. If the data are divided into this structure based on the MNIST dataset for each client, IID data means that data from 0–9 are distributed equally and held by the client. Non-IID data assumes federated learning in a state where a specific client has images labeled from 0–5 and another client has data from 6–9. It would be ideal if clients with specific data were composed only of clients with the same distribution based on the entire dataset. However, in reality, the data generated often have very different distributions for each client. As proven in various studies, when experiments are conducted with imbalanced data, the accuracy of federated learning is extremely low. This difference is illustrated in Figure 2.

The horizontal direction in Figure 2 shows the weight learning process of the AI model as learning progresses and indicates the number of client communications. The vertical direction shows the weight divergence of the model weights and is a graphical representation of the extent to which the model weights diverge as the communication rounds progress. As shown in Figure 2, when the local model was learned from non-IID data and averaged on the server, the difference in weights between the global and local models was large. In other words, in the case of IID data, if federated learning is performed based on multiple clients, there is a high possibility that it can converge to a specific model, and there is a high possibility that there will

be no major problems in learning the actual AI model. In the case of non-IID data, it may be difficult to converge to a specific AI model when simply learning the model and using only the weights of the learned model, such as FedAvg, and averaging the model. However, real-world data often have different data structures for each client, and there may be many cases in which only data with non-overlapping data labels are collected. Therefore, there have been continuous discussions on how to further tune these non-IID data or adjust algorithms for federated learning [33–36].

4 Federated Learning Based on an HPC Platform

On an HPC-based platform, users access HPC resources through a web platform, and various services are provided for HPC simulation execution and result analysis. To execute FL on such an HPC platform, it is necessary to define how to manage the execution resources for executing HPC simulations and the large number of AI models derived through federated learning (FL). Therefore, in this platform, we consider each user's simulation unit on the HPC platform as a unit for generating AI models and propose a method for storing, distributing, and sharing the generated AI models using blockchain. This is described in detail in Section 5.

4.1 AI Model Learning Based on an HPC Platform

Conducting AI learning on an HPC-based platform involves running a simulation by setting the resources and batch jobs in the simulation system within the HPC platform. As shown in Figure 3, users represented by each client using a web-based HPC simulation platform can create a model through AI learning based on the data they generate. This process can be run through batch jobs in the same manner as the HPC simulation running method, and each user can track or visualize the process. The AI model generated in this manner goes through a process of registering a learned AI model only for models that have reached certain conditions or thresholds through a validation and test process. The process of registering an AI model is stored through a smart contract within the blockchain. The AI model file is first registered in the IPFS system because there is a limit to the value that can be stored in a smart contract within the blockchain when registering a generated model. Smart contract requests and processing are made based on the CID (content ID) for the registered file. Figure 3 shows a diagram expressing the process of a client; that is, a user runs a simulation on an HPC web-based platform and

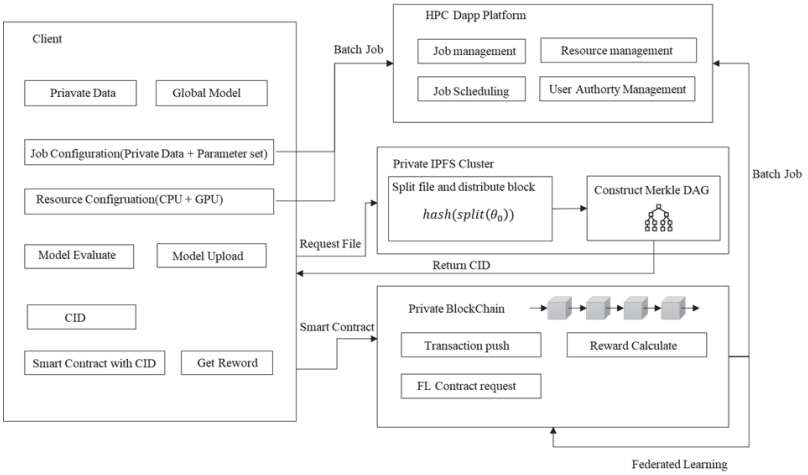


Figure 3 System architecture for federated learning on an HPC platform.

conducts a learning AI model with private data. After learning, the generated AI model is uploaded to the IPFS. Each user can obtain individual simulation results on a platform that can be visualized as an individual local environment, and the simulation results can be classified as private data that only each user can access. These private data can be provided as a single integrated service unit across the platform by undergoing an additional learning process to learn a single AI model integrated with other users without sharing other users' data, thereby creating a single global model.

The learning process at the client level is shown in Algorithm 1.

Various federated learning algorithms can be used by each client to perform AI learning locally. In this study, we added the FedProx method, which provides a mechanism that is not too biased toward specific data. The corresponding algorithm is shown in Algorithm 1. By sharing the entire model with the existing platform and adding additional variables to calculate the loss, the data distribution based on the global model was not overfitted to a specific dataset and maintained an equal data distribution for each client. Learning is performed based on a private dataset that can be used locally on the client side, as shown in Algorithm 1, and is enhanced by improving the loss through the addition of a proximal term. There are cases where each client has an unequal data distribution; for example, if one client has data only from digits 0 to 4 and another client has data only from digits 5 to 9 based on MINIST, the federated learning results of the two clients will drop to an accuracy of less than 50% if FedAvg is simply applied. Therefore,

learning should be performed using a local calculation formula that adds a regularization term to enable the weight values of the entire global model to be used for learning the local model, as shown in Algorithm 1. The dataset held by the client is private and is not shared with other users or clients. Thus, even if federated learning is performed, it is impossible to access the data from other clients' private datasets.

Algorithm 1: Deep learning with private data

Initialize server global model θ_g
$D_{private} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$, *current Client* C_0
Job confirugration with $R(r_{cpu}, r_{gpu}, r_m)$
batch job $J = \{C_0, R, D_{private}\}$
for *each epoch* ***do***
  for x_i, y_i ***in*** $D_{private}$
    Local loss $L_{local} = LossFunction(output, y_i)$
    Loss $L = L_{local} + \mu \sum_{k=1}^{\|D_{private}\|} \|(w_{local})_k - (w_{global})_k\|^2$
    $w^{t+1} \leftarrow w^t - \eta \frac{\delta L}{\delta w} L$
Return $\boldsymbol{\theta}_{C_0}$

As shown in Algorithm 1, the weights of the global model are required to initially proceed with learning on the client side. These weights are denoted as θ_g, and the value of the global model's weights is added to the local loss to facilitate learning, thereby flattening the overall data distribution. To proceed with a batch job on an HPC platform, it is necessary to set the resources that the user will use to simulate the batch job and specify the dataset to be used. Here, C_0 is a specific batch job, that is, the user who wants to proceed with learning, R means the resource configuration to be used for the batch job, and $D_{private}$ is the dataset used by the client. In other words, the dataset used by a specific client is accessible only to that client and is not visible to other users; therefore, the data cannot be modified. In addition, $\mu \sum_{k=1}^{\|D_{private}\|} \|(w_{local})_k - (w_{global})_k\|^2$, which is added to L_{local} used in local learning, calculates the distance difference between θ_g, which is the entire global model. $\theta_{C_0}$is the local model currently being trained that adds the value multiplied by μ that calculated the distance difference between θ_g and θ_{C_0}. μ is a hyperparameter that can be adjusted to a specific value for each client or simulation unit belonging to the federated learning group that trains the model.

The learned local model must exceed a certain threshold value to be registered as a transaction on the blockchain. The steps for registering it as

a transaction are shown in Algorithm 2. Based on the F1 score, a test was performed on the dataset of the model, as shown in Algorithm 2. If the F1 score of the learned model related to the dataset exceeds a certain threshold value compared with the global model, the learned model can be registered as a transaction. β is a constant value set so that a smart contract can be concluded as a transaction when it exceeds a certain value compared to the predicted result value of the global model.

Algorithm 2: Evaluation (F1 score based) local AI model

Initialize model with θ_g, θ_{C_0}
$N_g = \|D_g\|, N_{C_0} = \|D_{C_0}\|$
$k_g = \alpha \times N_g, k_{C_0} = \alpha \times N_{C_0}$
Initianlize Global Dataset $D_{ge} = RandomSample(D_g, k)$
Initianlize Local Private Dataset D_{pe}
$= RandomSample\ (D_p, k)$
$M_g = LoadM(\theta_g)$
$M_c = LoadM(\theta_{c_0})$
$F1(M_g) = 2 \times \frac{Precision(M_g) \times Recall(M_g)}{Precision(M_g) + Recall(M_g)}$
$F1(M_c) = 2 \times \frac{Precision(M_c) \times Recall(M_c)}{Precision(M_c) + Recall(M_c)}$
if $F1(M_c) \geq \beta \cdot F1(M_g)$
    return $(\theta_{c_0}, True)$
else
    return $(\theta_g, False)$

If the learned model on the client side exceeds a certain threshold value through the evaluation process of the new data model, the currently learned model can be registered. This process was uploaded to an internally constructed private IPFS, as shown in Figure 4.

If the uploaded file is extremely large, when a learning model is uploaded, the file is separated into blocks, and the block files are distributed and stored in the IPFS internal network. A Merkle tree is created to generate a content ID (CID) based on these distributed files, allowing access to the file using the generated CID. This IPFS network has the characteristic that, similar to a blockchain-based framework, once registered files cannot be falsified or deleted. It is often used together with a blockchain framework to store large files that are difficult to store in a blockchain alone owing to this characteristic. According to Algorithm 2, the file of a storable model is registered in the IPFS, and the client unit that performs federated learning on a working project basis can access the file. The data used for learning in

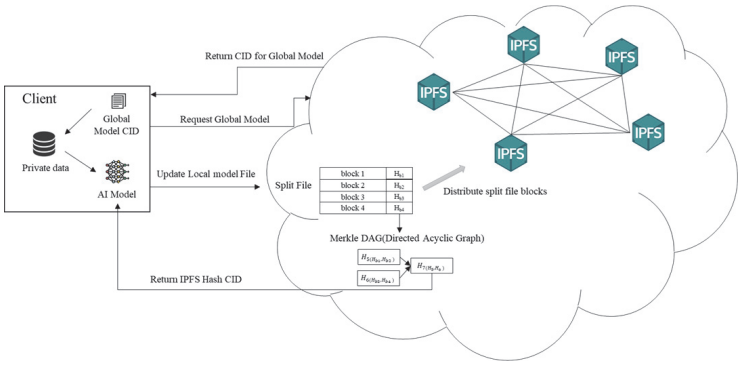


Figure 4 Upload process of the AI model.

each local unit is accessible only to the client that uploads or generates the data.

When a model is registered by the client unit and a CID for the registered model is received, the client registers the CID as a transaction in the blockchain based on the CID. The registration process is shown in Algorithm 3.

Algorithm 3: Contribution reward calculation

Initialize model weight θ_g, θ_{C0}
Set reward contant γ *for each push smart contract*
Initialize T is the number of transaction was created
Initialize α *as the rate that transaction*
Initialize n is the number of node
Initialize λ $istherateoftransactionspertime$ (t)
Initialize t is time
$TransactionT_i = SmartContract(\theta_g, \theta_{C_0})$
if T_i *is register on Consensus Algorithm*
  for b ***in*** *blocks*
    if b *include transaction from* C_0
      $C_{0,push} = C_{0,push} + count(T_{C_0})$
    end if
  end for
$R_{C_0} = \beta \cdot \lambda t((1 - e^{-\alpha n})C_{0,push}) + \gamma$
  return R_{C_0}

As the model is uploaded on a client-by-client basis, each time a model is uploaded, the transaction for the model uploaded by the client is registered in the blockchain. Various procedures are required to verify the registration

process of a specific client and authenticity of the registered data for the model registered on the blockchain. Because the blockchain is registered privately, the reward for the client is calculated based on the number of AI models registered by the client in the blockchain. Additionally, the reward considers the amount of time the client model has participated in the federated learning process. The reward for a specific client is calculated by considering the block mining speed (λ), the nodes participating in the blockchain network (n), the time when registration begins to learn the model (t), the propagation rate of the registered transaction (α), the transactions generated after model learning begins (T), etc. The weighted sum of the model is determined by considering the contribution of the client to learning the specific model and the contribution of the client that contributed to the entire block.

When a locally learned model is uploaded according to Algorithm 2, the model file is uploaded to the IPFS network and the corresponding CID is registered in the blockchain framework. The reward for the registered client is calculated when the transaction is registered. This calculation is performed because it is used as an additional weight value in the federated learning process. In the process of learning a model, a specific client learns the model based on a larger dataset and consumes considerable learning resources to register the model. The weight values of a specific user model are integrated with federated learning by applying the weight value. An additional weight value is applied to combine them into a single model, thereby applying the contribution of each client. The process proceeds with federated learning according to the contribution made to the learning process for the transaction in the block and the specific model, as shown in Algorithm 4.

Algorithm 4: Federation learning with blockchain

if b_i *is newly mined on the blockchain*
Initialize block b_i
  for *transaction* t *from* $T = \{t_1 = (\theta_l^g, \theta_l^{C_n}), t_2, \ldots, t_k\}$ *in* b_i
    $M_j : \theta_j = \frac{1}{|n|} \sum_{l=0}^{\|\theta^g\|} \theta_l^{C_n} \cdot R_{C_n}$
    $D_g \leftarrow$ *Download shared data with* θ^g
    $\theta_l^m = T_k = SmartContract\,(M_j : \theta_j, D_g)$
     $h_g \leftarrow$ *upload* $IPFS(M_j : \theta_j)$
  for *each Client* $k = 0, 1, 2, \ldots m$ *in parallel do*
    download M_j *with* $CID(h_g)$
  end for
 end for
end if

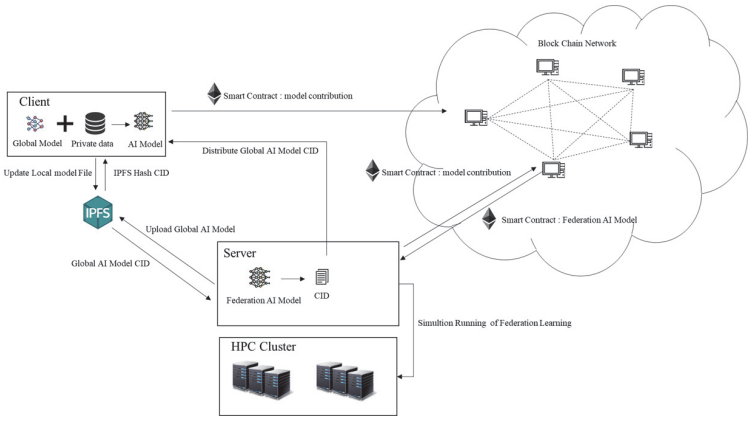


Figure 5 System architecture for federated learning.

This describes the process of creating a new model by combining the AI models included in transactions in the blockchain, as shown in Algorithm 4. It includes the relationship between clients and models for a specific model and the adjustment of the weights of federated learning according to the reward calculation of a specific client.

The entire process is represented graphically in Figure 5.

Simulations per client can be submitted and run in the form of batch jobs based on the HPC cluster, and the model can be trained using batch jobs based on the AI model. The resulting model file can be generated and registered on the IPFS. The process of registering the resulting model file as a transaction unit on the blockchain-based framework after the model verification task is completed is included. This process can be integrated into the simulated execution unit of each client as a specific project unit. Federated learning can be used as an algorithm that integrates the blockchain mining and federated learning processes based on the registered model and can also be submitted as a batch job to the HPC cluster.

4.2 AI Model Management based on Blockchain

Clients of the HPC platform can learn and register AI models based on the datasets they create. Through this process, clients learning the AI models of the same project unit can evaluate and register the locally learned models. Learning models registered as transaction units can be registered by multiple clients in the form of shared models. Assuming that learning is performed on a project unit to create a model based on a specific global model, various

clients participating in a project can register models corresponding to the blockchain as transaction units. As AI models cannot be modified or deleted once uploaded, they are registered after undergoing an evaluation and verification process. A verification process is also required for each client unit to register these models and proceed with federated learning through the registered learning models. In other words, because the learned model is registered and only the dataset learned locally is not registered, a verification process for the private dataset managed by each client unit is also required. Therefore, a verification process for clients that update the learned model is required when proceeding with federated learning. To perform federated learning on a specific AI model as a single AI model on the server side, a verification process for the private datasets held by each client is required. This process is described in Algorithm 5.

Algorithm 5: Evaluation and validation FL model

Download $M_j(\theta_j^g)$ *with CID* (h_g) *from IPFS*
Initialize weight θ_j^g
Initianlize Global Dataset $D_{ge} = RandomSample(D_g, k)$
Initianlize $V = \{v_1, v_2, \ldots, v_m\} \in C$
$C = \{c \mid c$ *use same model with* $\theta_j^g\}$
Initianlize $r = \{\theta_j^g, D_g\}$
Initianlize $v_m = \begin{cases} 1, & if\ v_m approves r \\ 0, & if\ v_m rejects r \end{cases}$
for v_m *in* V
  Initianlize Local Private Dataset $D_{pe} = RandomSample(D_p, k)$
  Initianlize Local Private Dataset $D_{pe} = RandomSample(D_p, k)$
  $M_g = LoadM(\theta_j^g)$
  $M_c = LoadM(\theta_{v_m})$
  $F1(M_g) = 2 \times \frac{Precision(M_g) \times Recall(M_g)}{Precision(M_g) + Recall(M_g)}$
  $F1(M_c) = 2 \times \frac{Precision(M_c) \times Recall(M_c)}{Precision(M_c) + Recall(M_c)}$
  if $F1(M_g) \geq \beta \cdot F1(M_c)$
   $v_m = 1$
  else
   $v_m = 0$
  end if
end for
vote aggregation and calculate approval rate $A_r = \frac{\sum_1^m v_m}{m}$
return *Decision* $(A_r) = \begin{cases} Approved, & if\ A_r \geq \gamma \\ Rejected, & if\ A_r < \gamma \end{cases}$

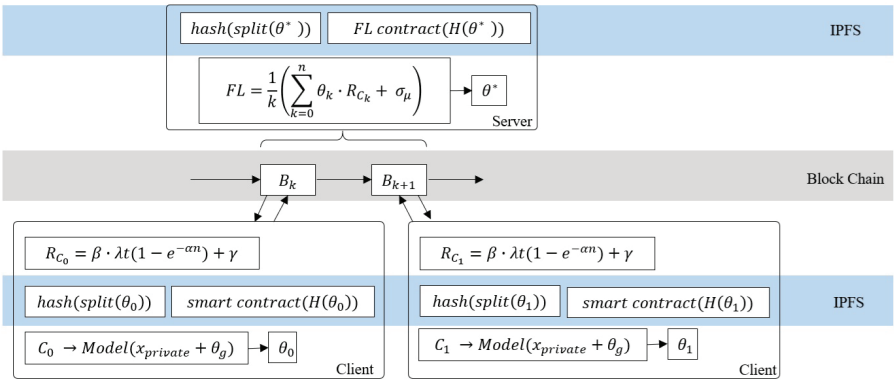


Figure 6 Federated learning with blockchain.

According to Algorithm 5, in order to verify the newly generated global model $M_j(\theta_j^g)$, verification of the clients who participated in creating the model is added. The model $M_j(\theta_j^g)$ is delivered to the clients participating in the project, and the clients who received the global model go through a verification process based on their private datasets. The new global model generated through federated learning using the dataset is verified on each client side and voted on whether to register the global model as a transaction. After verifying the verification process of all clients, the generated global model is registered as a transaction based on the results of the verification process. Therefore, the verification process, once again in the registration process, is a new transaction of the blockchain, including the entire verification process of the global model, which is finally generated through the entire federated learning process. The reliability of the global model can be ensured without sharing a local dataset.

Figure 6 shows a diagram describing the learning process of the client, the stored model, and the linkage process of the blockchain. Clients train models based on private datasets locally, verify the models, and register them as transactions in the blockchain. Clients receive rewards for the models during the process of registering transactions, and the weights of the models registered by multiple clients are combined based on the rewards to calculate the weights of the models. When a block is mined, a global model of federated learning for the block is created, and the created global model requires verification by clients again, and the verification process determines whether to register the global model through a transaction voting process.

Finally, the registered global model must be redistributed to all clients who contribute to the model learning. In other words, when learning is

performed on a newly created private dataset, it is performed based on a distributed global model.

5 Experiments on the Platform

The method proposed in this study is based on a web-based platform that creates tasks and processes large amounts of data as batch tasks for each user, based on large-scale calculations or AI tasks. Therefore, each user creates a project that targets a specific problem, and the data that each user has does not share. The data that users share on a project basis are based on the artificial intelligence model; only the artificial intelligence model with weights is uploaded to the IPFS system, and the AI model is written to the blockchain and shared. A model targeting a specific user is shared, federated learning is performed, and users participating in the target perform continuous model learning only for the available data. To learn for multiple clients, learning is performed based on the global model learned as a shared model.

5.1 Simulation on the Platform

The method for saving and distributing the model is illustrated in Figure 6. The method for saving and distributing each model can be set by using the input/output method within the platform. Each user can learn the model without sharing data with the other users. Resources that can be used based on the HPC platform can be set. A batch job can be executed during the federated learning process, and the simulation results can be checked through the platform. The results of FL learning within the platform are shown in Figures 7 and 8.

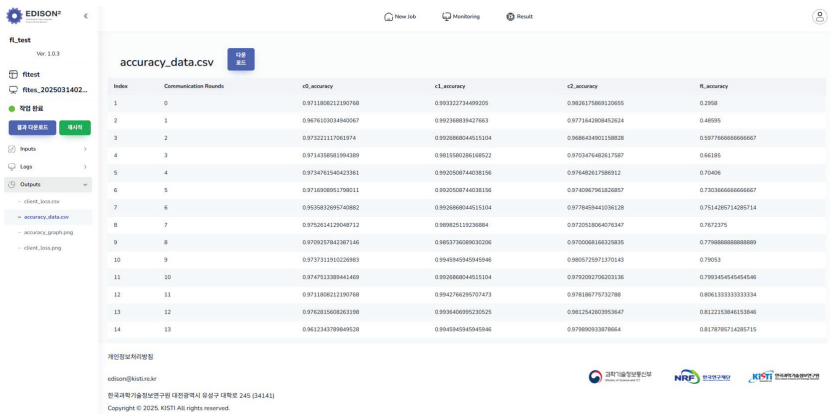


Figure 7 Batch job result.

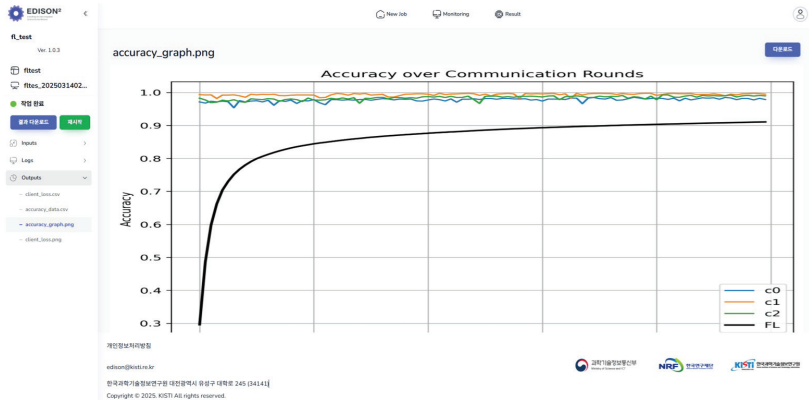


Figure 8 Federated learning result.

Table 2 HPC platform environment

Cluster	Nodes	GPU	CPU
Slurm	4	–	512
Kubernetes	8	H100*6	1600

Table 3 Software specification

Application	CPU	Memory	GPU	Running Time
FL_MNIST	1	16GB	3	9h17m14s
FL_COVID_UNET	1	16GB	3	13D22h54m11s

Each user can check the results screen as described above and check and share the result data to be executed as a batch job on the web-based platform screen. Consequently, the federated learning result model created as a global model can be shared, but the shared data for each user can be isolated and provided as a platform for creating the final model.

Table 2 below provides a brief summary of the execution environment and software used to run the experiments. Table 3 summarizes the names of applications registered and executed on the HPC platform, the resources used, and the execution time.

5.2 Evaluation of the FL Model

Federated learning can vary depending on how each user's model learning and the resulting data from the global model are conducted. The FL dataset evaluated in this study is based on federated learning using MNIST, a dataset that performs small calculations, and the COVID-19 medical dataset. The dataset was divided into random datasets and distributed to each client for

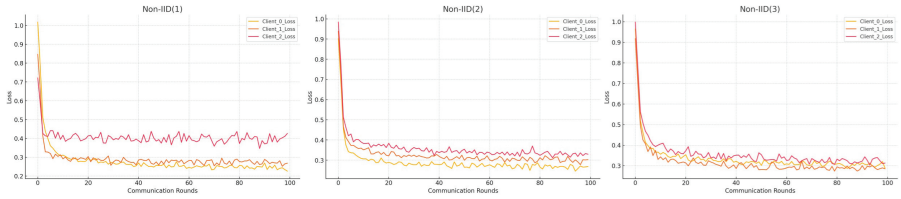


Figure 9 Federated learning with MNIST.

Table 4 Mnist client dataset

FL	Client Dataset	Rounds
Non-IID(1)	{(0,6,9,8), (1,4,7), (2,3,5)}	100
Non-IID(2)	{(0,6,9,1,2), (4,7,8,3,0), (5,2,3,9,4)}	100
Non-IID(3)	{(0,6,8,9,3), (1,4,7,9,0), (2,3,4,9,4)}	100

learning, and the client models learned in each round were combined to create a global model and shared for learning. The differences in model performance were analyzed depending on how the data were distributed to each client for learning.

The following graph shows the results of learning based on the MNIST dataset.

With learning with datasets that have overlapping or non-overlapping labels. The learning loss may differ depending on the degree of cross-labeling of the data. The separation of the dataset was defined as follows, and learning was conducted. In the case of non-IID(1), there are only clients that have fully independent datasets that do not share any data with each other, and each data point is divided into datasets that have characteristically similar labels and are learned. The learning rate of the data learning was set to $\eta = 0.01$ and $\mu = 0.1$. The second non-IID(2) clients were divided into datasets that each client shares one label and learned, and the learning rate and μ of the data being the same. Finally, the clients with non-IID(3) datasets were configured as clients with datasets that had labels, where three clients shared one label at the same time. Table 4 describes the client structure of each federated learning system.

Looking at the results in Figure 9, we can observe that in the case of non-IID(1), the loss results of the clients are not learned consistently, which means that the results of the client's learning are not maintained consistently, depending on the composition of the dataset. However, in the case of non-IID(2) or non-IID(3), the datasets are not completely independent, but 10–20% of the data is shared by each client and we can see that the learning

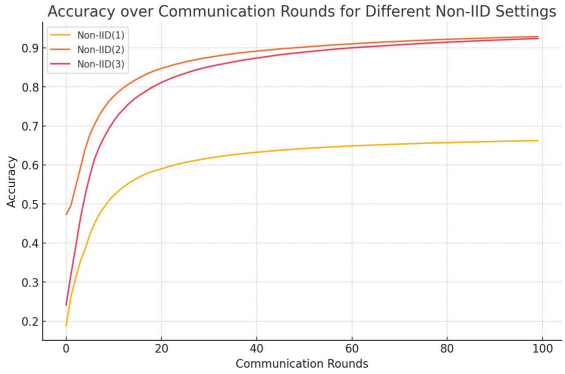
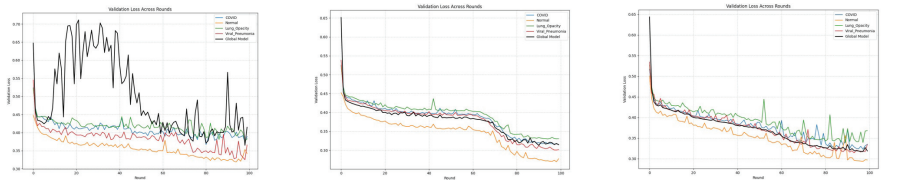


Figure 10 Global model for federated learning.



(a) Dataset with Fully Independent (b) Dataset with 10% data sharing on each client (c) Dataset with 20% data sharing on each client

Figure 11 Federated learning with COVID-19.

direction converges consistently while learning each datum. Figure 9 shows the results of the calculation of the learning loss of the global dataset for the dataset learned locally by each client. A graph of the global model is shown in Figure 10.

As shown in Figure 10, the final prediction accuracy of the global model for non-IID(1) was 0.662443. This is a much lower result than the structure of other non-IID(1), and it can be seen that when federated learning is performed between clients with completely independent data, the performance is very low. In addition, non-IID(2) shows an accuracy of 0.928702, and non-IID(3) shows an accuracy of 0.923741, which confirms that even if the datasets shared by each client are not completely independent and only share approximately 10% of the data, they show almost similar performance. Figure 11 shows the results of the learning data based on the UNet model using the COVID-19 medical dataset.

For the COVID-19 dataset, it was classified into four types based on the medical data. It comprises medical datasets for COVID, normal, lung opacity, and viral pneumonia. The data included lung X-ray and mask images,

Table 5 COVID-19 100 round loss

FL	COVID	Normal	Lung Opacity	Viral Pneumonia	Global Model
(a)	0.389368	0.342236	0.383007	0.365981	0.414864
(b)	0.315631	0.277553	0.330349	0.301546	0.314726
(c)	0.335146	0.297009	0.368396	0.334472	0.318874

and UNet-based model learning was performed based on the medical data. Learning was conducted over a total of 100 rounds, and it was performed with three configurations of clients, as shown in Figure 11. In Figure 11(a), each client learns with an independent dataset based on each medical data image of COVID, normal, lung opacity, and viral pneumonia, and the global model of the data is represented by a black line graph. In the case of the learning model for each client, it was observed that the learning of each model did not progress normally when creating a global model. In federated learning, all the clients had different diseases or normal lung X-ray images, and learning was conducted based on the fact that there were no shared labels. Consequently, the learning of the global model showed worse results than those of the local client models. In case (b) of Figure 11, federated learning was conducted by adding a setting in which 10% of the data was shared by all clients in the client dataset configuration of (a). From the learning results, it can be confirmed that learning progresses in the direction of constant convergence for both the local and global models. In addition, in Figure 11(c), it can be confirmed that learning progresses more stably in the result graph.

Finally, when federated learning was performed for 100 rounds, the resulting loss data in Figure 11 can be expressed as shown in Table 5.

As shown in Table 5, in the case of federated learning based on clients with completely independent datasets the accuracy of the global model was low. In addition, the learning graph shows unstable loss changes and severe data bias between clients, making it difficult to integrate model learning. In case (b), even with only 10% of the common data, it showed a more stable learning curve than the previous learning model, confirming that the generalization performance of the model improved. In addition, when the shared data increased to 20%, an overall stable learning curve was observed, confirming that the performance of federated learning improved as the similarity of the data distribution among the clients increased.

The results of predicting the lung images based on the final learned model as a result of federated learning are shown in Figures 12, 13, and 14. These are the results of images predicted using models learned based on the same randomly selected COVID, normal, lung opacity, and viral pneumonia

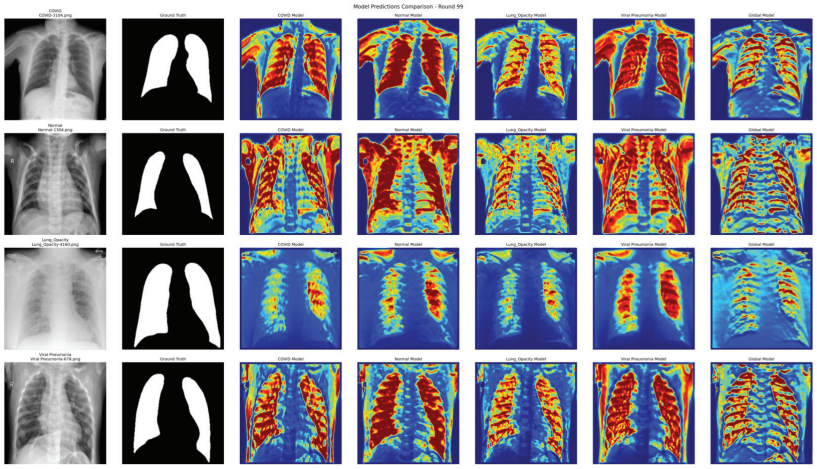


Figure 12 Visualized 11(a) model.

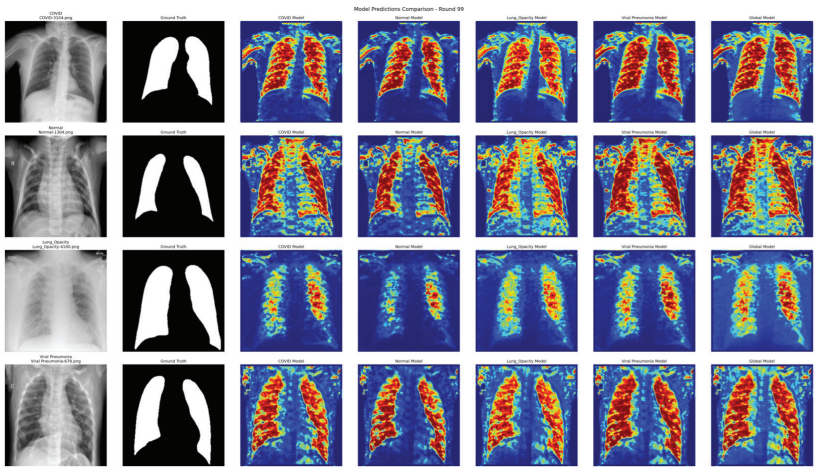


Figure 13 Visualized 11(b) model.

images. In each image, red and yellow indicate the parts that the model judged to be important. In the case of Figure 12, most models focused well on the lung area and generated predicted images. The COVID model and lung opacity model are located on both the border and center of the lung, while the viral pneumonia model showed strong responses to both the left and right lungs, capturing patterns similar to bilateral pneumonia. In Figure 13, in the case of the COVID model, attention is focused on the upper lobe of the left

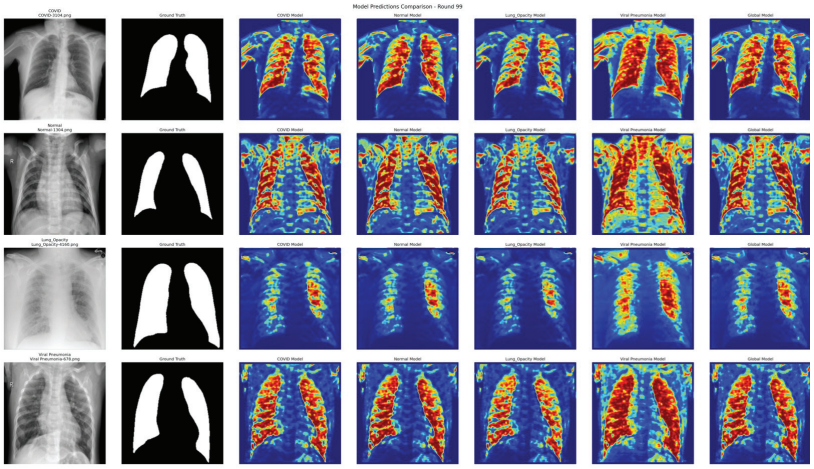


Figure 14 Visualized 11(c) model.

lung, which is an uneven area. In the case of the normal model, it responds uniformly overall but shows excessively activated light. In addition, the global model appeared to form the most stable prediction range by combining the attention areas of various models. Figure 14 is similar to Figure 13 and shows the overall stable and refined prediction model. The viral pneumonia model generated very similar patterns in the left and right lungs, while the COVID and lung opacity models showed stronger attention to the lower part of the lungs, which may indicate higher sensitivity to infectious lesions. The global model expressed a stable prediction range overall; however, in the case of lung opacity, it seemed that predicting the concentrated area of the lungs was the most difficult.

6 Conclusion

In this study, we proposed a federated learning framework that maximizes the sharing and performance of models while ensuring the privacy of personal data based on simulation data in the field of computational science and engineering. In particular, this system, which operates on an HPC-based platform, learns models without sharing data generated in a local environment and secures the authenticity and integrity of the models through a trust-based distributed model storage and sharing system that combines blockchain and IPFS.

The FedProx algorithm, among the federated learning algorithms that can effectively solve the problem of data imbalance, was introduced to minimize the performance degradation that may occur owing to data deviation between clients. In addition, the performance evaluation, registration, and reward systems of the model were automated based on blockchain smart contracts to quantitatively measure the contributions of participating clients and reflect the results of federated learning, thereby enabling more precise global model integration.

As a result of conducting experiments based on the MNIST and COVID-19 datasets, it was confirmed that model performance was significantly reduced in a non-IID environment where data between clients were completely independent; however, the learning stability and performance of the model were greatly improved with just 10–20% data sharing. This is an important result because most real-world data environments have a non-IID structure, suggesting that the data sharing structure must be considered when designing a federated learning framework in the future. The framework proposed in this paper can be expanded to various fields such as simulation-centered science and engineering research, as well as medical, financial, and IoT environments, where privacy is important. It shows the potential to be used as a base platform for a decentralized AI learning system in the future by securing blockchain-based security and traceability.

In future studies, we plan to perform additional improvements to various federated learning algorithms and optimization techniques to further enhance learning efficiency, conduct comparative experiments on various federated learning algorithms, improve the user interface, and verify scalability for the actual service commercialization of the platform.

Acknowledgement

This research was supported by the Ministry of Science and ICT through the National Research Foundation of Korea (NRF), under the Digital Convergence R&D Platform Program (No. NRF-2022M3C1A6090416), and by the Global TOP Strategic Research Group Program of the National Research Council of Science & Technology (No. GTL24031-700).

References

[1] Shokri, Reza, and Vitaly Shmatikov, "Privacy-preserving deep learning." Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, 2015, pp. 1310–1321.

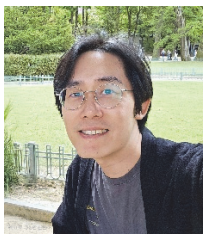
[2] Ryffel, Theo, et al., "A generic framework for privacy preserving deep learning." arXiv preprint arXiv:1811.04017, 2018, pp. 1–5.
[3] Tanuwidjaja, Harry Chandra, et al., "Privacy-preserving deep learning on machine learning as a service—a comprehensive survey." IEEE Access 8, 2020, pp. 167425–167447.
[4] Aono, Yoshinori, et al., "Privacy-preserving deep learning via additively homomorphic encryption." IEEE Transactions on Information Forensics and Security 13, no. 5, 2017, pp. 1333–1345.
[5] Kaplan, Jared, et al., "Scaling laws for neural language models." arXiv preprint arXiv:2001.08361, 2020, pp. 1–30.
[6] Shen, Jiahao, et al., "Blockchain-based distributed multi-agent reinforcement learning for collaborative multi-object tracking framework." IEEE Transactions on Computers, 2023, pp. 778–788.
[7] Qiu, Xiaoyu, et al., "Online deep reinforcement learning for computation offloading in blockchain-empowered mobile edge computing." IEEE Transactions on Vehicular Technology 68, no. 8, 2019, pp. 8050–8062.
[8] Gadiraju, Divija Swetha, V. Lalitha, and Vaneet Aggarwal, "An optimization framework based on deep reinforcement learning approaches for PRISM blockchain." IEEE Transactions on Services Computing 16, no. 4, 2023, pp. 2451–2461.
[9] Chowdhury, Sujit, Arnab Mukherjee, and Raju Halder, "FedRLChain: Secure federated deep reinforcement learning with blockchain." IEEE Transactions on Services Computing, 2023, pp. 3865–3878.
[10] Cam, Nguyen Tan, and Vu Tuan Kiet, "FlwrBC: Incentive mechanism design for federated learning by using blockchain." IEEE Access, 2023, pp. 107855–107866.
[11] Goh, Eunsu, et al., "Blockchain-enabled federated learning: A reference architecture design, implementation, and verification." IEEE Access, 2023, pp. 145747–145762.
[12] Han, Boyuan, et al., "Research on blockchain-based decentralized federated learning." 2023 International Conference on Computer Applications Technology (CCAT). IEEE, 2023, pp. 23–29.
[13] Nilsson, Adrian, et al., "A performance evaluation of federated learning algorithms." Proceedings of the Second Workshop on Distributed Infrastructures for Deep Learning, 2018, pp. 1–9.
[14] Zhou, Yuhao, Qing Ye, and Jiancheng Lv, "Communication-efficient federated learning with compensated Overlap-FedAvg." IEEE Transactions on Parallel and Distributed Systems 33, no. 1, 2021, pp. 192–205.

[15] Casella, Bruno, and Samuele Fonio, "Architecture-based FedAvg for vertical federated learning." Proceedings of the IEEE/ACM 16th International Conference on Utility and Cloud Computing, 2023, pp. 1–6.
[16] Mills, Jed, Jia Hu, and Geyong Min, "Communication-efficient federated learning for wireless edge intelligence in IoT." IEEE Internet of Things Journal 7, no. 7, 2019, pp. 5986–5994.
[17] Li, Tian, et al., "Federated optimization in heterogeneous networks." Proceedings of Machine Learning and Systems 2, 2020, pp. 429–450.
[18] Nguyen, Hung T., et al., "Fast-convergent federated learning." IEEE Journal on Selected Areas in Communications 39, no. 1, 2020, pp. 201–218.
[19] Zahri, Sofia, Hajar Bennouri, and Ahmed M. Abdelmoniem, "An empirical study of efficiency and privacy of federated learning algorithms." arXiv preprint arXiv:2312.15375, 2023, pp. 1–7.
[20] Geng, Jiahui, et al., "Improved gradient inversion attacks and defenses in federated learning." IEEE Transactions on Big Data, 2023, pp. 839–850.
[21] Kholod, Ivan, et al., "Open-source federated learning frameworks for IoT: A comparative review and analysis." Sensors 21, no. 1, 2020, pp. 167.
[22] Wei, Wenqi, et al., "Gradient-leakage resilient federated learning." 2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS). IEEE, 2021, pp. 797–807.
[23] Li, Zengpeng, Vishal Sharma, and Saraju P. Mohanty, "Preserving data privacy via federated learning: Challenges and solutions." IEEE Consumer Electronics Magazine 9, no. 3, 2020, pp. 8–16.
[24] Wang, Yingcheng, Songtao Guo, and Dewen Qiao, "FedSG: Subgraph federated learning on multiple non-IID graphs." 2023 19th International Conference on Mobility, Sensing and Networking (MSN). IEEE, 2023, pp. 504–511.
[25] Asad, Muhammad, Ahmed Moustafa, and Takayuki Ito, "FedOPT: Towards communication efficiency and privacy preservation in federated learning." Applied Sciences 10, no. 8, 2020, pp. 1–17.
[26] Ahmed, Syed Thouheed, et al., "FedOPT: Federated learning-based heterogeneous resource recommendation and optimization for edge computing." Soft Computing, 2024, pp. 1–12.
[27] Wang, Hongyi, et al., "Federated learning with matched averaging." arXiv preprint arXiv:2002.06440, 2020, pp. 1–16.

[28] Sannara, E. K., et al., "A federated learning aggregation algorithm for pervasive computing: Evaluation and comparison." 2021 IEEE International Conference on Pervasive Computing and Communications (PerCom). IEEE, 2021, pp. 1–10.
[29] Ek, Sannara, et al., "Evaluation of federated learning aggregation algorithms: Application to human activity recognition." Adjunct Proceedings of the 2020 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2020 ACM International Symposium on Wearable Computers, 2020, pp. 638–643.
[30] Ek, Sannara, et al., "Evaluation and comparison of federated learning algorithms for human activity recognition on smartphones." Pervasive and Mobile Computing 87, 2022, pp. 14453–14464.
[31] Sun, Yan, et al. "Efficient federated learning via local adaptive amended optimizer with linear speedup." IEEE Transactions on Pattern Analysis and Machine Intelligence 45.12, 2023, pp. 14453–14464.
[32] Reddi, Sashank, et al., "Adaptive federated optimization." arXiv preprint arXiv:2003.00295, 2020. pp. 1–9.
[33] Zhao, Yue, et al., "Federated learning with non-IID data." arXiv preprint arXiv:1806.00582, 2018, pp. 1–12.
[34] Li, Xiang, et al., "On the convergence of FedAvg on non-IID data." arXiv preprint arXiv:1907.02189, 2019, pp. 1–26.
[35] Ma, Xiaodong, et al., "A state-of-the-art survey on solving non-IID data in federated learning." Future Generation Computer Systems 135, 2022, pp. 244–258.
[36] Zhu, Hangyu, et al., "Federated learning on non-IID data: A survey." Neurocomputing 465, 2021, pp. 371–390.
[37] Calvin, Christophe, and France Boillod-Cerneux. "HPC and Data: When Two Becomes One." Turbulence and Interactions. Cham: Springer International Publishing, 2018, pp. 14–25.
[38] Yuhang Chen, Wenke Huang, and Mang Ye, "Fair federated learning under domain skew with local consistency and domain diversity", In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2024, pp. 12077–12086.
[39] Wang, Zheng, et al. "Federated Learning with Domain Shift Eraser." Proceedings of the Computer Vision and Pattern Recognition Conference(CVPR), 2025, pp. 4978–4987.

Biographies

Yejin Kwon received a bachelor's degree in computer science from Dankook University in 2011 and a master's degree in computer science from Dankook University in 2014. She is currently working as a Researcher at the Department of Supercomputing Acceleration Research, Division of National Supercomputing R&D, Korea Institute of Science and Technology Information. Her research areas are HPC based platforms, AI based scheduling, blockchain, and software engineering.



Jeongcheol Lee received a bachelor's degree in computer engineering from Chungnam National University in 2008, and a philosophy of doctorate degree in computer engineering from Chungnam National University in 2014. He conducted postdoctoral research at the University of California, Los Angeles (UCLA) from 2015 to 2017. He is currently working as a Principal Researcher at the Department of Supercomputing Acceleration Research, Division of National Supercomputing R&D, Korea Institute of Science and Technology Information. His research areas include artificial intelligence, computational science, cloud platforms, simulation workflows, and wireless networks.

Youngbom Park received a bachelor's degree in computer science from Sogang University in 1985, a master's degree in computer engineering from NY Polytechnic University in 1987, and a philosophy of doctorate degree in computer science from NY Polytechnic University in 1991. He is currently working as a Professor at the Department of Software Engineering, Faculty of Engineering, Dankook University. His research areas include artificial intelligence, blockchain, deep learning, and software engineering.