Software: Practice and Experience

WILEY

RESEARCH ARTICLE OPEN ACCESS

Federated Learning Architectures: A Performance Evaluation With Crop Yield Prediction Application

Anwesha Mukherjee[1,2] | Rajkumar Buyya[1]

[1]Quantum Cloud Computing and Distributed Systems (qCLOUDS) Laboratory, School of Computing and Information Systems, The University of Melbourne, Victoria, Australia | [2]Department of Computer Science, Mahishadal Raj College, Mahishadal, West Bengal, India

Correspondence: Anwesha Mukherjee (anweshamukherjee2011@gmail.com)

Received: 2 August 2024 | **Revised:** 16 January 2025 | **Accepted:** 30 January 2025

Funding: This work is partially supported by an ARC Discovery Project (DP240102088) and ARC LIEF.

Keywords: federated learning | prediction accuracy | response time | training time

ABSTRACT

Introduction: Federated learning has become an emerging technology in data analysis for IoT applications.
Methods: This paper implements centralized and decentralized federated learning frameworks for crop yield prediction based on Long Short-Term Memory Network and Gated Recurrent Unit. For centralized federated learning, multiple clients and one server are considered, where the clients exchange their model updates with the server that works as the aggregator to build the global model. For the decentralized framework, a collaborative network is formed among the devices either using ring topology or using mesh topology. In this network, each device receives model updates from the neighboring devices and performs aggregation to build the upgraded model.
Results: The performance of the centralized and decentralized federated learning frameworks is evaluated in terms of prediction accuracy, precision, recall, F1-Score, and training time. The experimental results show that >93% prediction accuracy is achieved using the centralized and decentralized federated learning-based frameworks. The results also show that using centralized federated learning, the response time can be reduced by ~75% than the cloud-only framework.
Conclusion: Centralized and decentralized federated learning architectures show good performance in terms of prediction accuracy and loss. The training time, including communication for both case studies, is also not very high, as observed from the results. Further, as no raw data is shared, the data privacy is protected.
Finally, the future research directions of the use of federated learning in crop yield prediction are proposed.

1 | Introduction

Agriculture is an important sector that has a huge impact on the economy of most of the countries. The conventional farming practices depend on manual decision-making on crop harvesting, irrigation, etc., that suffers from improper selection of crops, inefficient utilization of lands, water resources, etc. To overcome these problems, smart agriculture and farming practices come into the scenario, where the Internet of Things (IoT) plays a significant role [1, 2]. In IoT-based smart agricultural systems, IoT devices are used for soil and environmental parameters' data collection, and then the data is analyzed for decision-making [3, 4]. For data analysis, machine learning (ML) and deep learning (DL) are used, and the cloud servers are used for storing and analyzing the large volume of data. However, data storage and analysis inside the cloud require huge amounts of IoT data transmission to the cloud, which requires seamless network connectivity and high network bandwidth. However, agricultural lands are mainly located at rural areas, where the network bandwidth may not be high, as well as seamless network connectivity may not be available. Hence, data transmission from IoT devices to the cloud is a challenge. Further, entire data transmission from IoT devices to the cloud causes high network traffic, and storage and analysis of the huge volume of data increase cloud overhead, latency, etc. To address these issues, edge computing and fog computing have come [4, 5]. In edge computing, the resources are placed at the edge of the network to reduce the latency and computation overhead on the cloud. In fog computing, the intermediate devices, e.g., switch, router, etc., process data to reduce latency and overhead on the cloud. Nevertheless, the concern regarding data security and privacy still remains. Moreover, the soil data, environmental data, and climate and weather conditions of various geographical regions are different, and the user may not like to transmit and store the data over the cloud due to privacy issues. Hence, personalized local models and an efficient global model are required to perform accurate predictions and utilize the agricultural resources efficiently. To utilize the edge devices for local data analysis and for collaborative training to develop a global model, federated learning (FL) [6–8] comes into the scenario.

FL is a learning approach that allows collaborative training with the coordination of multiple devices and a central server without sharing individual datasets [7, 9]. In an edge-cloud-based FL, the edge devices serve as the clients, and the cloud server acts as the central server for collaborative training [5]. In an FL-based framework [7], the central server serves as the aggregator that at first creates an initial global model with learning parameters. Each of the clients downloads the current model from the server, computes it own model updates using its local dataset, and offloads the local update to the server. The server receives local updates from all clients and develops an improved global model. The clients download the global update from the server, compute their local updates again, and offload the updates to the server. This process is continued until the global training is finished. The principal advantages of FL are enhanced data privacy, low-latency and learning quality enhancement [7]. As no data is shared, privacy is protected [9, 10]. Further, an enhanced version of the global model is created through collaborative training. As local data analysis takes place, the latency is reduced.

According to networking structure and data partitioning, the FL algorithms are classified into several categories. Based on data partitioning, FL algorithms are classified into three types [7]: Vertical FL, Horizontal FL, and federated transfer learning. In vertical FL systems, the clients have datasets with the same sample space but different feature spaces. In horizontal FL systems, the clients have datasets with the same feature spaces but different sample space. In federated transfer learning, the clients have datasets with different feature spaces as well as different sample space. According to the networking structure, FL algorithms are divided into two categories: Centralized Federated Learning (CFL) and Decentralized Federated Learning (DFL). In a CFL system, all the clients train a model in parallel using their local datasets. Then, the clients send the trained parameters to the server. The server aggregates model parameters after receiving them from all clients and builds the updated global model. The clients get the updated model parameters from the server for the next training round. After the server finishes the global model training, each of the clients has the same global model as well as its personalized local model. However, communication with the server may not always be available. In such a case, DFL can be adopted. In DFL, all the clients form a collaborative network. For each of the communication rounds, the clients use their local datasets for local training. After that, each client performs aggregation based on model updates received from the neighbor nodes.

In this article, we focus on the use of FL in the field of agriculture, especially crop yield prediction. Though various frameworks on FL exist, their use in agricultural data analysis was not much explored. Further, the implementation of FL in agricultural data analysis was not provided in detail in existing frameworks. Agricultural data of different geographical regions vary due to variations in soil, environment, and weather parameters. Further, the data of each farm is confidential because the soil compositions of agricultural lands differ, and production highly relies on the soil, environment, and weather parameters. In such a circumstance, a cloud-only system for data analysis compromises data privacy. Moreover, agricultural lands are located in rural regions, and therefore, seamless data transmission to the cloud servers is difficult. To deal with these issues, the use of FL is recommended along with a detailed implementation of both centralized and decentralized architectures in this paper. The collaborative learning also helps to build a model with good prediction accuracy. The paper provides a study of the centralized and decentralized FL architectures with a case study of crop yield prediction.

1.1 | Motivation and Contributions

Crop yield prediction is an important domain of smart agriculture, where ML is used for data analysis [11]. As farmers' information is shared as well as soil and environmental parameters' data analysis takes place, privacy is a major issue. In such a case, the use of a conventional cloud-only framework for data analysis using ML raises concerns regarding data privacy, latency, connectivity interruption due to poor network connectivity inside the agricultural lands, etc. To address these issues, the objective of the paper is to explore the use of FL for crop yield prediction. The major contributions of the paper are:

- The use of FL in farming practices is discussed, and then an experimental case study is performed to analyze the performance of CFL and DFL in crop yield prediction. We consider a scenario where different numbers of devices perform collaborative training using CFL and DFL. Long Short-Term Memory (LSTM) Network and Gated Recurrent Unit (GRU) are used as the underlying approaches for both the CFL and DFL mechanisms.
- To implement CFL, a client-server paradigm is developed using socket programming, and multiple clients are handled by the server in the conducted experiment. For transmission of model updates *MLSocket* is used. For aggregation, Federated Averaging (FedAvg) is used. The performance of the CFL-based framework has been evaluated in terms of prediction accuracy, precision, recall, F1-Score, and training time. The experimental results show that the CFL-based framework has better prediction accuracy and lower response time than the cloud-only framework, where

- the cloud server analyzes the data after receiving it from the client.
- To implement the DFL, a collaborative network is formed using mesh topology or ring topology. In the conducted experiment, each node receives model updates from the neighbour nodes (the neighbor nodes depend on the selected topology), and performs aggregation to build the upgraded model. The performance of the nodes in both the topologies is evaluated in terms of prediction accuracy, precision, recall, F1-Score, and training time.
- Finally, the research challenges with CFL and DFL-based frameworks in crop yield prediction are highlighted in this paper.

The rest of the paper is organized as follows: Section 2 briefly discusses the existing literature on FL, smart agriculture, and crop yield prediction. Section 3 illustrates the use of CFL and DFL in crop yield prediction. An experimental case study is presented in Section 4 on the use of CFL and DFL in crop yield prediction. Section 5 demonstrates another case study to analyze the performance of CFL and DFL. Section 6 explores the future research directions of FL in crop yield prediction. Finally, Section 7 concludes the paper.

The list of acronyms used in this article is listed in Table 1.

2 | Related Work

Crop yield prediction and recommendation is a crucial area of IoT-based smart farming practices [1, 2]. There are several research works carried out on the use of ML and DL in crop yield prediction. In [12], the authors explored the use of several ML algorithms such as KNN, DT, RF, XGBoost, and SVM for crop yield prediction. In [13], the authors used MLP, decision tables, and JRip for crop yield prediction. In [14], KNN was used for crop yield prediction. In [15], the authors used several ML approaches such as DT, SVM, KNN, LGBM, and RF for crop yield prediction. The LSTM, Bi-LSTM, and GRU-based framework was used in [16] for data analysis to predict crop yield. For crop yield prediction, MLR with ANN was used in [17]. In [18], Bi-LSTM was used for data analysis, and for better network connectivity, the use of a small cell with computation ability was explored. However, none of the existing approaches adopted FL in their frameworks. The major disadvantages of the conventional ML-based framework are compromise with data privacy as data sharing takes place with the cloud for analysis, the requirement of high network bandwidth that may not be available in rural regions, high response time, high network traffic, huge overhead on the cloud server, etc. To address all these issues, FL can be adopted in crop yield prediction.

The concept of FL relies on local data analysis, collaborative training, and generating global as well as personalized models. As no individual dataset is shared and a distributed learning is performed, data privacy is protected [7]. Further, due to its distributed nature, FL can be adopted in various IoT applications, including healthcare, agriculture, transportation systems, etc. The use of FL in IoT was elaborated in [7]. The use of FL in a fog computing environment was explored in [19]. In [20], for distributed data analytics, FL and transfer learning were adopted to propose an intelligent microservices-based framework for IoT applications. The use of FL in agriculture was explored in a few research works. In [5], CFL was used for soil health monitoring for irrigation decision-making. The authors used CFL based on LSTM and DNN in their work. In [21], FL was used for soybean yield prediction using deep residual network-based regression models for risk management in agricultural production. In [22], FL was used for intrusion detection in IoT-based agricultural systems. For yield forecasting, FL was used in [23]. For efficient data sharing in the agri-food sector, the use of FL was discussed in [24]. For crop classification, FL was used by [25]. The authors adopted CFL for crop classification based on Gaussian NB in [25]. As we observe, the use of FL in crop yield prediction was explored in a few existing works, and most of them focused on the use of CFL. In this paper, we provide an experimental study of both the CFL and DFL in crop yield prediction.

In Table 2, the existing works are compared with the proposed FL-based framework for crop yield prediction. As we observe from the table, most of the existing works rely on conventional ML/DL-based framework, and compared to the existing CFL-based framework, this work explores the use of both CFL

TABLE 1 | Acronyms with full forms.

Acronyms	Full form
ANN	Artificial Neural Network
ML	Machine Learning
DL	Deep Learning
FL	Federated Learning
CFL	Centralized Federated Learning
DFL	Decentralized Federated Learning
IoT	Internet of Things
IoAT	Internet of Agricultural Things
LSTM	Long Short-Term Memory Network
Bi-LSTM	Bidirectional Long Short-Term Memory Network
FedAvg	Federated Averaging
KNN	K-Nearest Neighbours
DT	Decision Tree
RF	Random Forest
XGBoost	Extreme Gradient Boosting
SVM	Support Vector Machine
MLP	MultiLayer Perceptron
LGBM	Light Gradient Boosting Machine
GRU	Gated Recurrent Unit
MLR	Multiple Linear Regression
NB	Naive Bayes
DNN	Deep Neural Network
RNN	Recurrent Neural Network
P2P	Peer-to-Peer
FTL	Federated Transfer Learning
AWS	Amazon Web Services

TABLE 2 | Comparison of our work with existing crop yield prediction frameworks.

Work	Classifier	Multi-crop dataset is used	CFL is used	DFL is used	Training time is measured	Response time is measured
[12]	RF, DT, KNN XGBoost, SVM	✓	✗	✗	✗	✗
[13]	MLP, Decision Table JRip	✓	✗	✗	Measured model build time	✗
[14]	KNN	✓	✗	✗	✗	✗
[15]	DT, SVM, KNN LGBM, RF	✓	✗	✗	✗	✗
[16]	LSTM, Bi-LSTM GRU	✓	✗	✗	✗	✗
[18]	Bi-LSTM	✓	✗	✗	Measured model build time	Measured latency
[25]	Gaussian NB	✓	✓	✗	✗	✗
Our work	LSTM	✓	✓	✓	✓	✓

and DFL (with mesh as well as ring-based frameworks) in crop yield prediction and determines the training time as well as response time.

3 | Federated Learning in Crop Yield Prediction

In Section 1, we have briefly discussed CFL and DFL. For crop yield prediction, both approaches can be used. In an IoT-based crop yield prediction framework, the IoT devices collect data on soil and environmental parameters such as temperature, humidity, rainfall, soil pH, Nitrogen, Phosphorus, Potassium level, etc. The collected IoT data is processed inside the cloud servers. However, for data privacy protection, to reduce latency and deal with poor network connectivity inside the rural regions containing the agricultural lands, FL is adopted in IoAT. The integration of FL with an IoT-based crop yield prediction framework permits local data analysis inside the edge devices that work as the clients. For data analysis we have used LSTM and GRU in this work. To capture the temporal dependencies and retain the sequential nature of the soil and environmental data, LSTM and GRU are considered. The mathematical notations used in this work are defined in Table 3.

GRU is an RNN that uses gates to update the hidden state selectively at each time step. The GRU has two gates: The reset gate and the update gate. The reset gate is used to determine how much of the previous hidden state to forget and the update gate is used to determine how much of the new input to use for hidden state update. The reset gate is mathematically presented as:

$$\theta_t = \sigma(weight_\theta * [h_{t-1}, x_t]) \tag{1}$$

where σ denotes the sigmoid function.

The update gate is presented as:

$$\mu_t = \sigma(weight_\mu * [h_{t-1}, x_t]) \tag{2}$$

The candidate hidden state is presented as:

$$h_{t'} = tanh(weight_h * [\theta_t * h_{t-1}, x_t]) \tag{3}$$

where $weight_h$ denotes the weight.

The hidden state is presented as:

$$h_t = (1 - \mu_t) * h_{t-1} + \mu_t * h_{t'} \tag{4}$$

LSTM is an upgraded version of RNN that considers a memory cell that is controlled by an input gate, a forget gate, and an output gate. LSTM maintains a chain-like structure that has four neural networks and different memory blocks referred to as cells. To learn long-term dependencies, the gates play significant roles by retaining or discarding information in a selective manner. For the short-term memory, a hidden state is maintained by the LSTM network that is updated depending on the previous hidden state, input, and the current state of the memory cell.

In LSTM, the forget gate controls which information will be removed from the memory cell and is mathematically expressed as:

$$\phi_t = \sigma(weight_\phi \ldots [h_{t-1}, x_t] + bias_\phi) \tag{5}$$

where σ denotes the sigmoid function.

The input gate is used to add information to the memory cell and is mathematically expressed as:

$$\zeta_t = \sigma(weight_\zeta \ldots [h_{t-1}, x_t] + bias_\zeta) \tag{6}$$

where a sigmoid function is used to regulate the information and filter the values to retain. After that, the *tanh* function is used to create a vector having all possible values from h_{t-1} and x_t, as follows:

$$\hat{C}_t = tanh(weight_C \ldots [h_{t-1}, x_t] + bias_C) \tag{7}$$

TABLE 3 | Mathematical notations with definitions.

Notation	Definition
θ_t	Reset gate
μ_t	Update gate
ϕ_t	Forget gate
ζ_t	Input gate
λ_t	Output gate
$weight_\theta$	Weight matrix of reset gate
$weight_\mu$	Weight matrix of update gate
$weight_\phi$	Weight matrix of forget gate
$weight_\zeta$	Weight matrix of input gate
$weight_\lambda$	Weight matrix of output gate
$bias_\phi$	Bias for forget gate
$bias_\zeta$	Bias for input gate
$bias_\lambda$	Bias for output gate
h_t	Present hidden state
h_{t-1}	Previous hidden state
x_t	Current input
$\hat{C}_t$	Candidate value
C_t	Cell state
C_{t-1}	Previous cell state
$Data_c$	Data of client c
B	Batch size
η	Learning rate
N_b	Number of batches
N_e	Number of epochs
N_r	Number of rounds
m_c	Model update of client c
N_c	Number of connected clients
f_c	Fraction of clients participating in CFL
m_s	Model parameter of the server
m_{final}	Final global model update
m_0	Initial model parameters of a node in DFL
m_p	Model update of node p in DFL
P	Set of nodes in the DFL framework
N_p	Number of neighbors nodes in DFL
$\mathcal{L}$	Loss function
α	True positive
β	True negative
γ	False positive
ρ	False negative
$\mathcal{A}$	Accuracy
$\mathcal{P}$	Precision
$\mathcal{R}$	Recall
$\mathcal{F}$	F1-Score

where $weight_C$ denotes the weight matrix and $bias_C$ denotes the bias.

Finally, the regulated values are multiplied with the values of the vector to get the information to be added to the memory cell, as follows:

$$C_t = \phi_t \odot C_{t-1} + \zeta_t \odot \hat{C}_t \tag{8}$$

The output gate that extracts the useful information from the current memory cell state as the output, is mathematically expressed as follows:

$$\lambda_t = \sigma(weight_\lambda \ldots [h_{t-1}, x_t] + bias_\lambda) \tag{9}$$

Firstly, using the *tanh* function, a vector is generated. After that, a sigmoid function is used to regulate the information and filter the values to retain using the inputs h_{t-1} and x_t. Finally, the regulated values are multiplied with the values of the vector to be sent as an output as well as input to the next cell. As LSTM is able to capture long-term dependencies, LSTM performs well in sequence prediction tasks, time series, etc.

3.1 | CFL-Based Framework

In our CFL-based system, all the clients get the initial model parameters from the server, train their individual models using their local datasets, and then transmit the model updates to the server node. The server node works as the aggregator that receives model updates from all the clients and performs aggregation to update the global model accordingly. Here, for aggregation, we use FedAvg. The updated global model is sent to the participating clients. Hence, at the end of the process, each client has its personalized local model and the global model. The client and server-side algorithms are stated in Algorithms 1 and 2, respectively. In the algorithms, c represents a client and S represents the server. Algorithm 1 presents the steps of the client-side process. The clients get connected with the server and get model parameters. Each client splits its local dataset into batches and

ALGORITHM 1 | Client-side algorithm.

```
Input: $Data_c, N_b, N_e$
Output: $m_c$
1:  $Function\ Update(Data_c, N_b, N_e)$:
2:  $PData_c \leftarrow Preprocess(Data_c)$
3:  while $connected\ with\ S$ do
4:      $Train(m_c \leftarrow getmodelparameters())$
5:  end while
6:  $SaveParameters(m_c)$
7:  $Function\ Train(m_c)$:
8:  $N_b \leftarrow Split(PData_c, B)$                ▷ Split data into $N_b$ batches
9:  for $e = 0\ to\ N_e - 1$ do
10:     for $b = 1\ to\ N_b$ do
11:         $m_c^{e+1} \leftarrow m_c^e - \eta \nabla m_c^e$      ▷ $\nabla m_c^e$ represents the gradient
12:     end for
13: end for
14: $m_c \leftarrow m_c^{N_e}$
15: $sendmodelupdate(m_c)$                 ▷ Send model update to $S$
```

ALGORITHM 2 | Server-side algorithm.

```
Input: $N_c, f_c, N_r$
Output: $m_{final}$
1: Function $Collect(N_c, N_r)$:
2: $ConnectedClients \leftarrow []$
3: while $(length(ConnectedClients) \neq N_c)$ do
4:     $listen()$
5:     $acceptconnection()$
6: end while
7: $FedAvg()$
8: Release clients
9: Function $FedAvg()$:
10: $m_s^0 \leftarrow InitModel()$                ▷ initial model is generated
11: for $r = 1$ to $N_r$ do
12:     $M_r \leftarrow Subset(max(f_c * N_c, 1), \}\}randome)$
13:     $MU \leftarrow []$
14:     for $c \in M_r$ do
15:         $m_c^r \leftarrow getmodelupdate(c)$     ▷ Get model update from client $c$ at round $r$
16:         $MU.append(m_c^r)$   ▷ Append model update of client $c$
17:     end for
18:     $m_s^{r+1} \leftarrow \frac{1}{|M_r|} \sum_{c=1}^{|M_r|} \cdot m_c^r$
19:     $sendtoclients(m_s^{r+1})$
20: end for
21: $m_{final} \leftarrow m_s^{N_r+1}$
```

performs local model training. After the training, the client saves the updated model parameters and sends the model update to the server. Algorithm 2 presents the steps of the server-side process. The server accepts connections for N_c clients. After receiving model updates from all the clients, the server stores the model updates in MU. After receiving model updates from all the clients, the server performs aggregation using $FedAvg()$ function after each round, and sends the updated model parameters to the clients. This process is repeated for the number of rounds considered in the FL process. The pictorial representation of CFL is presented in Figure 1, where N_c clients participate in a collaborative training process with the server that works as the aggregator to aggregate the model updates received from the clients to build the global model.

In CFL, the server is the aggregator, and it distributes the model updates with the clients. The clients have their personalized models along with the global model update. Each of the clients performs data analysis locally through a collaborative training process without sharing the data. Hence, privacy is protected, as well as through collaborative training, prediction accuracy is enhanced. The time complexity of the CFL process depends on the time complexity of model initialization, local model training, exchange of model updates, and aggregation. The time complexity of model initialization is given as $O(1)$. The time complexity of local model training is given as $O(N_r \cdot N_e \cdot N_b \cdot m_c)$. The time complexity of exchanging model updates is given as $O(N_r \cdot N_c \cdot (m_c + m_s))$. The time complexity of aggregation is given as $O(N_r \cdot N_c \cdot m_c)$.

Though there are several benefits, the CFL has some limitations. As the server performs as the aggregator, good network connectivity with the server is highly desirable. However, many applications do not have the provision of seamless connectivity with the server. Further, the overhead on the server is very high because the aggregation takes place inside the server. Further, sharing model updates by all the clients with the server may raise a concern regarding security. To address these limitations, DFL has come.

3.2 | DFL-Based Framework

In the case of crop yield prediction, the data collection takes place in the rural regions, where the network connectivity is usually poor. In that case, the communication with the cloud server is a major issue. Therefore, if connectivity with the cloud server is poor, DFL can be used by the edge devices for collaborative training purposes. In DFL, the clients form a network among themselves and perform collaborative learning. Here, each node is a learner as well as contributor. In our work, we have considered two types of DFL frameworks where the clients form a network either using ring or mesh topology. The ring-based network is referred to as P2P network also, where each peer exchanges its model updates with two neighbor nodes. In the case of the mesh-based network, each node exchanges its model updates with the rest of the nodes in that network. The DFL process for ring-based and mesh-based frameworks is stated in Algorithms 3 and 4, respectively, where p denotes a node and P denotes the set of nodes in the formed network. In Algorithms 3 and 4, j denotes a neighbor node. In a ring-based framework, each node has a preceding node (p_{pre}) and a succeeding node (p_{suc}), which act as its neighbors. In the mesh-based network, each node is directly connected with the rest of the nodes. Hence, in a mesh-based framework, each node has the rest of the nodes as its neighbors. As each node communicates with the rest of the nodes, the number of communications will be $(|P|(|P|-1))/2$, where $|P|$ denotes the number of nodes in the formed network. In both of the mesh and ring-based DFL frameworks, each node splits its own local dataset and performs local model training. After training, each node sends and receives model updates to and from the neighbors. The pictorial representations of DFL using ring topology and mesh topology are presented in Figures 2 and 3, where four nodes form a network using ring topology and mesh topology, respectively.

We observe from Algorithm 3 that in the ring-based DFL approach, a node p sends and receives model updates to and from its preceding node (p_{pre}) and succeeding node (p_{suc}). In the mesh-based DFL approach as presented in Algorithm 4, a node p sends and receives model updates to and from the rest of the nodes of the network. As the model updates are received from the neighbor nodes, the received updates are appended to store all the received updates in M_{recv}. Finally, aggregation takes place based on the received updates from the nodes in both the ring-based and mesh-based frameworks. The time complexity of the DFL-based framework depends on the time complexity of model initialization, local model training, exchanging model updates, and aggregation. The time complexity of model initialization is given as $O(1)$. The time complexity of local model training is given as $O(N_r \cdot N_e \cdot N_b \cdot m_p)$. The time complexity of exchanging model updates is given as $O(N_r \cdot N_p \cdot (m_p + m_j))$. The time complexity of aggregation is given as $O(N_r \cdot N_p \cdot m_j)$, where $1 \leq j \leq N_p$.

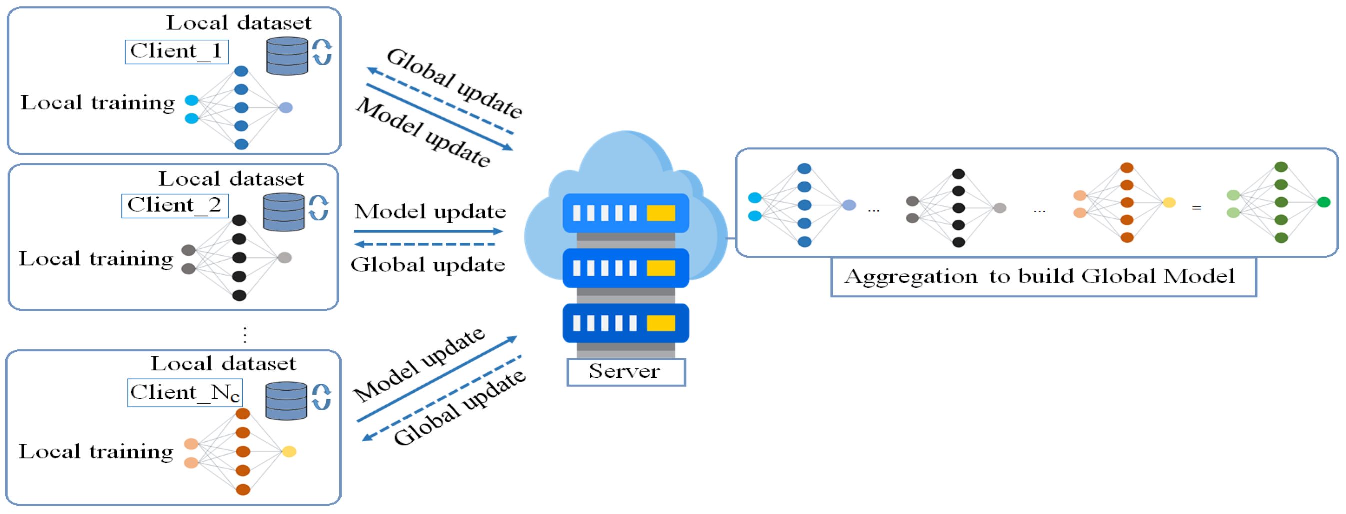

FIGURE 1 | The CFL process.

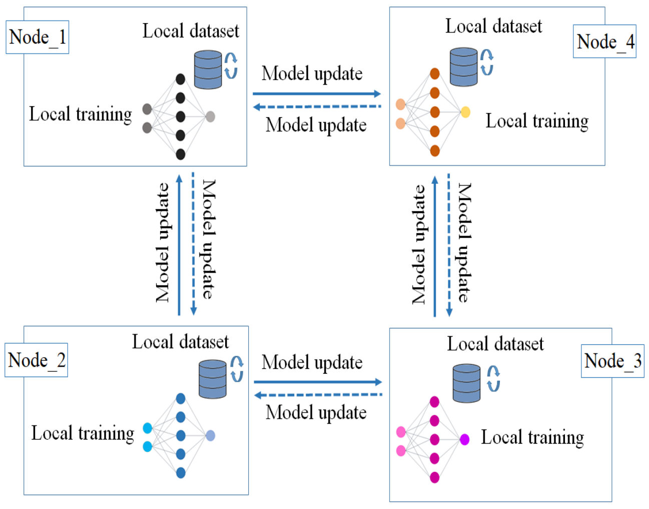

FIGURE 2 | DFL framework using ring topology.

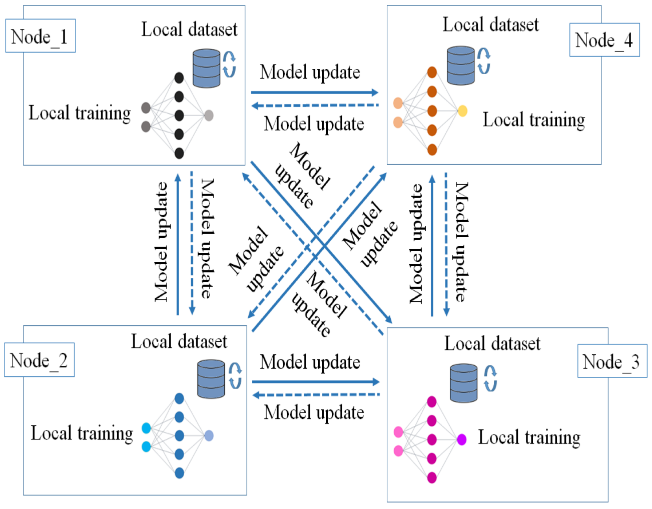

FIGURE 3 | DFL framework using mesh topology.

If a DFL framework contains three nodes, then the number of exchanges of model updates will be the same for both the mesh and ring-based networks. However, for *number of nodes* >= 4, the results will be different as the number of model updates exchanged differ for the mesh and ring-based networks.

3.3 | Proof of Convergence in FL

For both the CFL and DFL approaches, the final objective is to minimize the global loss function $\mathcal{L}(m)$ that is mathematically defined as follows:

$$\mathcal{L}(m) = \frac{1}{K}\sum_{i=1}^{K}\mathcal{L}_i(m) \tag{10}$$

where K is the number of participating nodes, i.e., $K = N_c$ for CFL and $K = N_p$ for DFL, and $\mathcal{L}_i(m)$ is the local loss function at node i.

Definition 1. *Lipschitz Continuity*: $\mathcal{L}_i(m)$ is $\mathcal{L}$-Lipschitz continuous if there exists a constant $C_1 > 0$ such that $\forall m, \mathbf{w}$, we have

$$\mathcal{L}_i(m) \leq \mathcal{L}_p(\mathbf{w}) + \nabla\mathcal{L}_p(\mathbf{w})^T(m - \mathbf{w}) + \frac{C_1}{2}\|m - \mathbf{w}\|^2 \tag{11}$$

Definition 2. *Bounded Variance*: The variance of the stochastic gradients is bounded if there exists a constant $C_2^2 > 0$ such that $\forall m$:

$$\mathbb{E}[\|\nabla\mathcal{L}_i(m) - \nabla\mathcal{L}(m)\|^2] \leq C_2^2 \tag{12}$$

Definition 3. *Unbiased Gradients*: The stochastic gradients are unbiased estimates of the true gradients if $\forall m$:

$$\mathbb{E}[\nabla\mathcal{L}_i(m)] = \nabla\mathcal{L}(m) \tag{13}$$

ALGORITHM 3 | Model update in ring-based P2P framework.

```
Input: $Data_p, P, N_p, m_0, N_r$
Output: $m_p, \forall p \in P$
1: $Function\ Training(N_p, m_0)$:
2: for $p = 1\ to\ |P|$ do
3:     $m_p \leftarrow m_0$                      ▷ Initialize model parameters
4: end for
5: for $r = 1\ to\ N_r$ do
6:     for $p = 1\ to\ |P|$ do                   ▷ Local model training
7:         $PData_p \leftarrow Preprocess(Data_p)$
8:         $N_b \leftarrow Split(PData_p, B)$  ▷ Split data into $N_b$ batches
9:         for $e = 0\ to\ N_e - 1$ do
10:            for $b = 1\ to\ N_b$ do
11:                $m_p^{e+1} \leftarrow m_p^e - \eta \nabla m_p^e$   ▷ $\nabla m_p^e$ represents the gradient
12:            end for
13:        end for
14:    end for
15:    for $p = 1\ to\ |P|$ do      ▷ Exchange of model updates and aggregation
16:        $M_{recv} \leftarrow []$
17:        for $j \in \{p_{pre}, p_{suc}\}$ do
18:            Send $m_p$ to $j$   ▷ Send model parameters to node $j$
19:            $m_j^r \leftarrow getmodelupdate(j)$ ▷ Get model parameters from node $j$
20:            $M_{recv}.append(m_j^r)$   ▷ Append model parameters received from node $j$
21:        end for
22:        $m_p^{r+1} \leftarrow \sum_{j=1}^{N_p} m_j^r / N_p$   ▷ Aggregate model updates received from $p_{pre}$ and $p_{suc}$
23:    end for
24: end for
```

ALGORITHM 4 | Model update in mesh-based framework.

```
Input: $Data_p, P, N_p, m_0, N_r$
Output: $m_p, \forall p \in P$
1: $Function\ Training(N_p, m_0)$:
2: for $p = 1\ to\ |P|$ do
3:     $m_p \leftarrow m_0$                      ▷ Initialize model parameters
4: end for
5: for $r = 1\ to\ N_r$ do
6:     for $p = 1\ to\ |P|$ do                   ▷ Local model training
7:         $PData_p \leftarrow Preprocess(Data_p)$
8:         $N_b \leftarrow Split(PData_p, B)$  ▷ Split data into $N_b$ batches
9:         for $e = 0\ to\ N_e - 1$ do
10:            for $b = 1\ to\ N_b$ do
11:                $m_p^{e+1} \leftarrow m_p^e - \eta \nabla m_p^e$   ▷ $\nabla m_p^e$ represents the gradient
12:            end for
13:        end for
14:    end for
15:    for $p = 1\ to\ |P|$ do      ▷ Exchange of model updates and aggregation
16:        $M_{recv} \leftarrow []$
17:        for $j = 1\ to\ N_p$ do
18:            Send $m_p$ to $j$, where $j \in (P - p)$   ▷ Send model parameters to node $j$
19:            $m_j^r \leftarrow getmodelupdate(j)$, where $j \in (P - p)$   ▷ Get model parameters from node $j$
20:            $M_{recv}.append(m_j^r)$   ▷ Append model parameters received from node $j$
21:        end for
22:        $m_p^{r+1} \leftarrow \sum_{j=1}^{N_p} m_j^r / N_p$   ▷ Aggregate model updates received from all other nodes
23:    end for
24: end for
```

Assumption 1. *Smoothness*: The global loss function $\mathcal{L}(m)$ is smooth, i.e., there exists a constant $C_3 > 0$, such that $\forall m, \mathbf{w}$:

$$\mathcal{L}(m) \leq \mathcal{L}(\mathbf{w}) + \nabla\mathcal{L}(\mathbf{w})^T(m - \mathbf{w}) + \frac{C_3}{2}\|m - \mathbf{w}\|^2 \tag{14}$$

Assumption 2. *Strong Convexity*: The global loss function $\mathcal{L}(m)$ is strongly convex, i.e., there exists a constant $C_4 > 0$ such that $\forall m, \mathbf{w}$:

$$\mathcal{L}(m) \geq \mathcal{L}(\mathbf{w}) + \nabla\mathcal{L}(\mathbf{w})^T(m - \mathbf{w}) + \frac{C_4}{2}\|m - \mathbf{w}\|^2 \tag{15}$$

Lemma 1. *Gradient Bound: Under the assumptions of Lipschitz continuity and bounded variance, the gradient of $\mathcal{L}(m)$ is bounded*:

$$\mathbb{E}[\|\nabla\mathcal{L}(m)\|^2] \leq \frac{C_1}{K}\sum_{i=1}^{K}\|\nabla\mathcal{L}_i(m)\|^2 + \frac{C_2^2}{K} \tag{16}$$

Proof. By Lipschitz continuity of $\mathcal{L}_i(m)$:

$$\|\nabla\mathcal{L}_i(m)\|^2 \leq C_1^2\|m\|^2 \tag{17}$$

Taking the expectation and summing over all nodes:

$$\mathbb{E}[\|\nabla\mathcal{L}(m)\|^2] = \frac{1}{K^2}\sum_{i=1}^{K}\mathbb{E}[\|\nabla\mathcal{L}_i(m)\|^2] \leq \frac{C_1^2}{K^2}\sum_{i=1}^{K}\|m\|^2 + \frac{C_2^2}{K} \tag{18}$$

Thus,

$$\mathbb{E}[\|\nabla\mathcal{L}(m)\|^2] \leq \frac{C_1}{K}\sum_{i=1}^{K}\|\nabla\mathcal{L}_i(m)\|^2 + \frac{C_2^2}{K} \tag{19}$$

□

Theorem 1. *Under the Lipschitz continuity, bounded variance, unbiased gradients, smoothness, and strong convexity, the FL-based framework converges to a stationary point of $\mathcal{L}(m)$.*

Proof. *Step 1: Local Update Rule*: Each node i performs local updates using stochastic gradient descent for N_e local epochs. Let m_i^e denote the model parameters at node i at epoch e. Then,

$$m_i^{e+1} = m_i^e - \eta\nabla m_i^e \tag{20}$$

Step 2: Aggregation: After N_e epochs, each node exchanges its model update with other nodes in DFL and with the server in CFL. For CFL, the aggregation takes place at the server. After receiving the model update, the client trains with local dataset and sends the update to the server.

For CFL, the global model update at round *r* is defined as:

$$m^{r+1} = \frac{1}{K}\sum_{i=1}^{K} m_i^r \tag{21}$$

where $K = N_c$.

For DFL, each node aggregates the updates after receiving from neighbor nodes (N_p). Hence, the local model update at round *r* is defined as:

$$m_i^{r+1} = \frac{1}{N_p}\sum_{j=1}^{N_p} m_j^r \tag{22}$$

The global model update at round *r* is defined as:

$$m^{r+1} = \frac{1}{K}\sum_{i=1}^{K} m_i^{r+1} \tag{23}$$

where $K = N_p$.

Step 3: Bounding the Global Loss: Using the smoothness assumption, the change in $\mathcal{L}(m)$ is expressed as follows:

$$\mathcal{L}(m^{r+1}) \le \mathcal{L}(m^r) + \nabla\mathcal{L}(m^r)^{N_r}(m^{r+1} - m^r) + \frac{C_2}{2}\|m^{r+1} - m^r\|^2 \tag{24}$$

Taking the expectation over the stochastic gradients, we obtain

$$\mathbb{E}[\mathcal{L}(m^{r+1})] \le \mathcal{L}(m^r) - \frac{\eta}{K}\|\nabla\mathcal{L}(m^r)\|^2 + \frac{C_3\eta^2 C_2^2}{2K} \tag{25}$$

Now, summing this inequality over N_r rounds, we get

$$\sum_{r=1}^{N_r}\mathbb{E}[\mathcal{L}(m^{r+1}) - \mathcal{L}(m^r)] \le -\frac{\eta}{K}\sum_{r=1}^{N_r}\|\nabla\mathcal{L}(m^r)\|^2 + \frac{C_3\eta^2 C_2^2 N_r}{2K} \tag{26}$$

After rearranging terms, we find

$$\frac{1}{N_r}\sum_{r=1}^{N_r}\mathbb{E}[\|\nabla\mathcal{L}(m^r)\|^2] \le \frac{\mathcal{L}(m^1) - \mathcal{L}(m^{N_r+1})}{\eta N_r} + \frac{C_3\eta C_2^2}{2K} \tag{27}$$

As $N_r \to \infty$, the term $\frac{\mathcal{L}(m^1)-\mathcal{L}(m^{N_r+1})}{\eta N_r}$ approaches zero, ensuring the following

$$\lim_{N_r\to\infty}\frac{1}{N_r}\sum_{r=1}^{N_r}\mathbb{E}[\|\nabla\mathcal{L}(m^r)\|^2] = 0 \tag{28}$$

This demonstrates that the global model update *m* converges to a stationary point of $\mathcal{L}(m)$. □

3.4 | Performance Metrics

In the next section, we have analyzed the performance of both the CFL and DFL-based frameworks in crop yield prediction in terms of prediction accuracy, precision, recall, F1-Score, and average training time per round.

The accuracy of a model is determined as follows:

$$\mathcal{A} = \frac{\alpha + \beta}{\alpha + \beta + \gamma + \rho} \tag{29}$$

The precision of a model is determined as follows:

$$\mathcal{P} = \frac{\alpha}{\alpha + \gamma} \tag{30}$$

The recall of a model is determined as follows:

$$\mathcal{R} = \frac{\alpha}{\alpha + \rho} \tag{31}$$

The F1-Score of a model is determined as follows:

$$\mathcal{F} = 2 * \frac{\mathcal{P} * \mathcal{R}}{\mathcal{P} + \mathcal{R}} \tag{32}$$

We have already discussed the time complexity of CFL and DFL, where we observe that the time consumption depends on the time consumed for model initialization, training, exchange of model updates, and aggregation.

Therefore, in the CFL-based approach, the training time (T_{CFL}) of the server is determined as the sum of the time consumption for model initialization (T_{init_1}), model training (T_{train_1}), exchanging updates with the clients (T_{ex_1}), and aggregation (T_{agg_1}), given as

$$T_{CFL} = T_{init_1} + T_{train_1} + T_{ex_1} + T_{agg_1} \tag{33}$$

In the DFL-based framework, the training time (T_{DFL}) of a node is determined as the sum of the time consumption for model initialization (T_{init_2}), model training (T_{train_2}), exchanging updates with the neighbor nodes (T_{ex_2}), and aggregation (T_{agg_2}), given as

$$T_{DFL} = T_{init_2} + T_{train_2} + T_{ex_2} + T_{agg_2} \tag{34}$$

The response time (T_{resp}) is determined as the difference between the timestamp of receiving the result after prediction and the timestamp of submitting the request for prediction, given as

$$T_{resp} = T_{recvr} - T_{subr} \tag{35}$$

where T_{recvr} denotes the timestamp of receiving the predicted result, and T_{subr} denotes the timestamp of submitting the request.

4 | Performance Evaluation

This section describes the implementation[1], the experimental setup used for analysis, and then analyzes the performance of CFL and DFL based on the experimental setup.

4.1 | Implementation

A design and implementation of the architecture of CFL shown in Figure 1 is presented in Figure 4 along with detailing interaction between the FL clients and the global server. In Figures 2 and 3, we have presented the architecture of DFL using ring and mesh topology, respectively. Here, we present the implementation diagrams of DFL using ring and mesh topology with four nodes in Figures 5 and 6, respectively. For implementation, we use Python language. *Tensorflow* is used along with LSTM and GRU supported by it. To build the client-server model and communication over the network, socket programming is used. For the transfer of model updates, we have used *MLSocket*. For secure communication, Secure Shell protocol is used. The considered dataset is split, and assigned to the clients as the local datasets, and to the server as the global dataset. As the first and second dense layer activation function *ReLU* is used in the LSTM-based framework. In both LSTM and GRU-based frameworks, *Softmax* is used as the Output layer activation function, *Categorical Crossentropy* is used as the Loss function, and as the optimizer, *Adam* is used.

4.2 | Experimental Setup

The experiment was conducted in the CLOUDS lab at the University of Melbourne. We created *sixteen instances (H1 to H16)* over the AWS-based Australian academic *RONIN Cloud Platform*. The configuration of each instance is:

- 4 GB RAM
- 2 vCPUs
- 100 GB SSD

Among these instances *one instance* (H3) was selected as the *server machine*, and other *fifteen instances* (H1, H2, H4–H15) served as the *client machines*. The performance of CFL and DFL in crop yield prediction was analyzed in terms of prediction accuracy and average training time per round. For experimental analysis, we used the dataset[2], containing 2200 samples of 22 different classes (rice, maize, pigeonpeas, chickpea, mothbeans, mungbean, kidneybeans, blackgram, lentil, jute, grapes, pomegranate, watermelon, mango, muskmelon, orange, papaya, banana, apple, coconut, cotton, coffee). There are seven features: Nitrogen (N), Phosphorous (P), Potassium (K), temperature, pH, humidity, and rainfall, which were considered as the input, and the crop was considered as the output. The learning rate was considered 0.001. The number of local epochs was considered 30 in both CFL and DFL.

4.3 | CFL in Crop Yield Prediction

In CFL, we performed three case studies: (i) Scenario 1: Five instances as client machines and one instance as the server machine, (ii) Scenario 2: Ten instances as client machines and one instance as the server machine, and (iii) Scenario 3: Fifteen instances as client machines and one instance as the server machine. We considered $f_c = 1$. Each of the client machines had its local dataset, and the server machine had the global dataset. The client machines shared their model updates with the server machine. The server machine performed aggregation and updated the global model. The global model update was then shared with the clients. The maximum number of rounds was considered 10.

4.3.1 | Accuracy, Precision, Recall, and F1-Score

The prediction accuracy, precision, recall, and F1-score of the global model using LSTM for the considered three scenarios are presented in Figure 7. As we observe from the figure, the global model using LSTM has achieved a prediction accuracy of 0.97 in scenario 3. In scenarios 1 and 2, the global model using

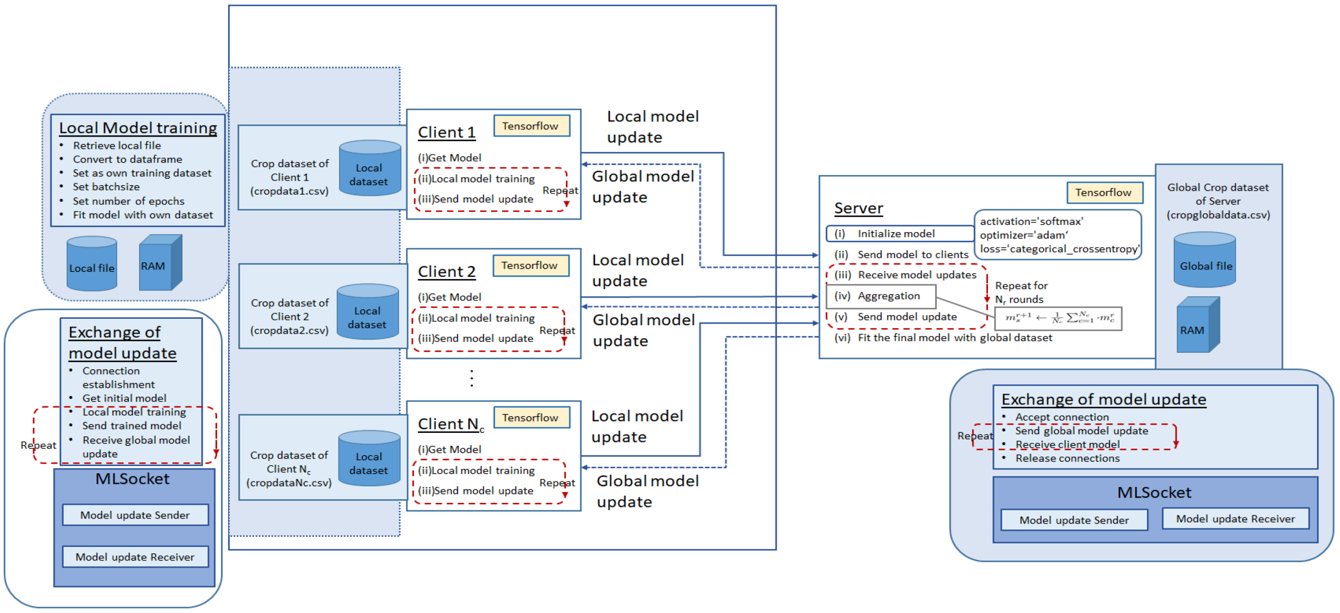

FIGURE 4 | The experimental implementation diagram of CFL-based crop yield prediction.

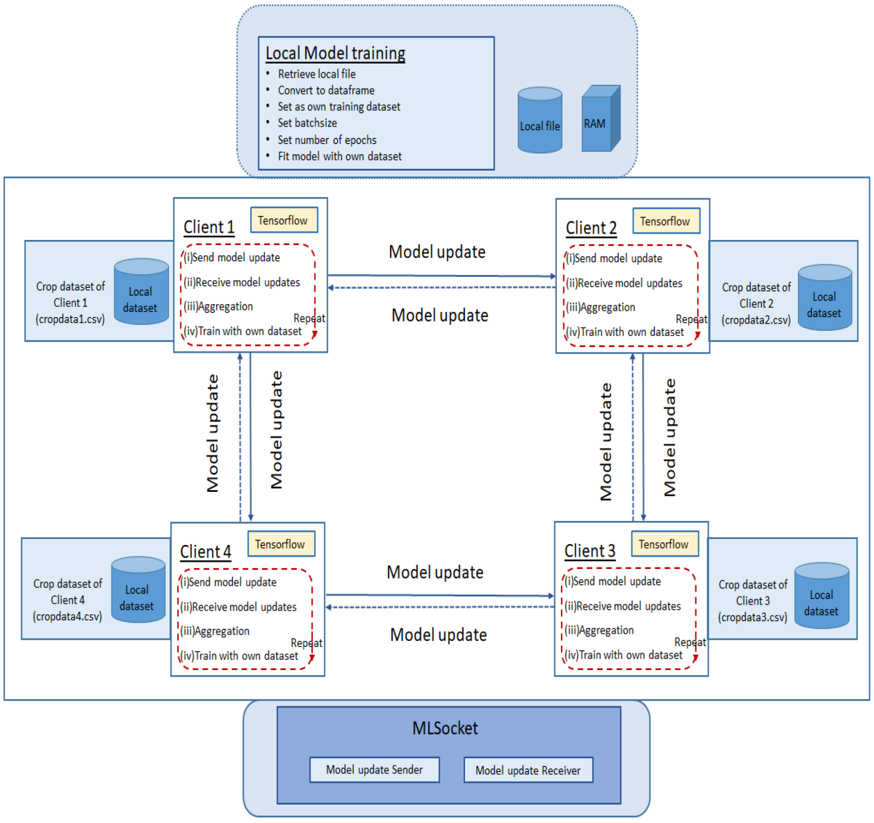
Local Model training
Retrieve local file
Convert to dataframe
Set as own training dataset
Set batchsize
Set number of epochs
Fit model with own dataset
Local file
RAM
Client 1
Client 2
Client 3
Client 4
Tensorflow
(i)Send model update
(ii)Receive model updates
(iii)Aggregation
(iv)Train with own dataset
Repeat
Crop dataset of Client 1 (cropdata1.csv)
Crop dataset of Client 2 (cropdata2.csv)
Crop dataset of Client 3 (cropdata3.csv)
Crop dataset of Client 4 (cropdata4.csv)
Local dataset
Model update
MLSocket
Model update Sender
Model update Receiver

FIGURE 5 | The experimental implementation diagram of DFL-based crop yield prediction using ring topology.

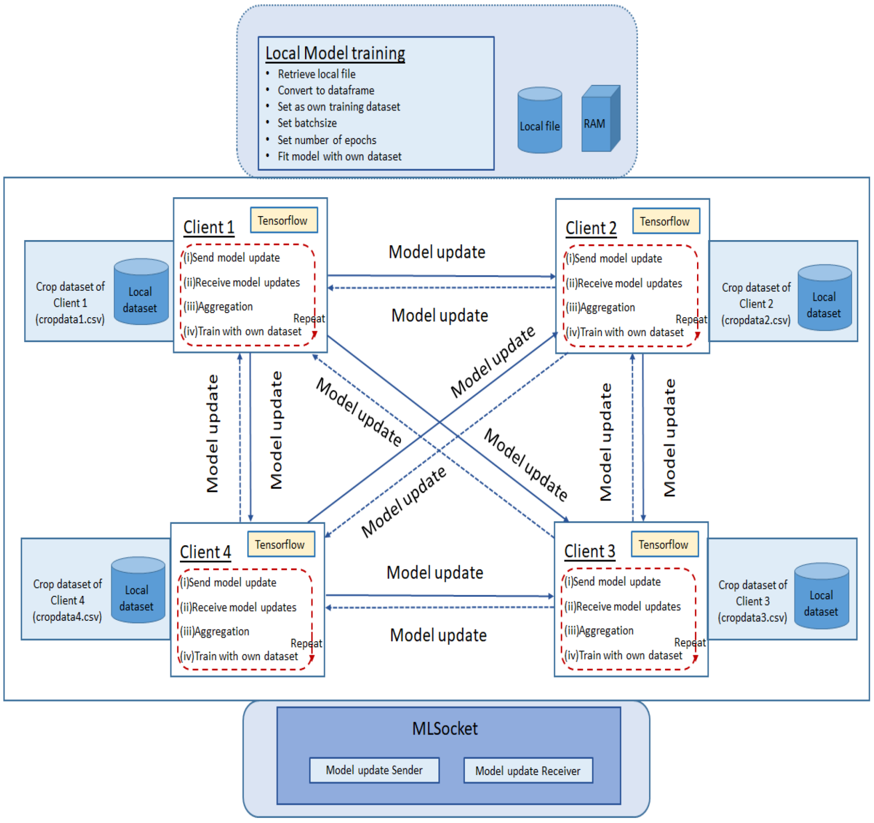
Local Model training
Retrieve local file
Convert to dataframe
Set as own training dataset
Set batchsize
Set number of epochs
Fit model with own dataset
Local file
RAM
Client 1
Client 2
Client 3
Client 4
Tensorflow
(i)Send model update
(ii)Receive model updates
(iii)Aggregation
(iv)Train with own dataset
Repeat
Crop dataset of Client 1 (cropdata1.csv)
Crop dataset of Client 2 (cropdata2.csv)
Crop dataset of Client 3 (cropdata3.csv)
Crop dataset of Client 4 (cropdata4.csv)
Local dataset
Model update
MLSocket
Model update Sender
Model update Receiver

FIGURE 6 | The experimental implementation diagram of DFL-based crop yield prediction using mesh topology.

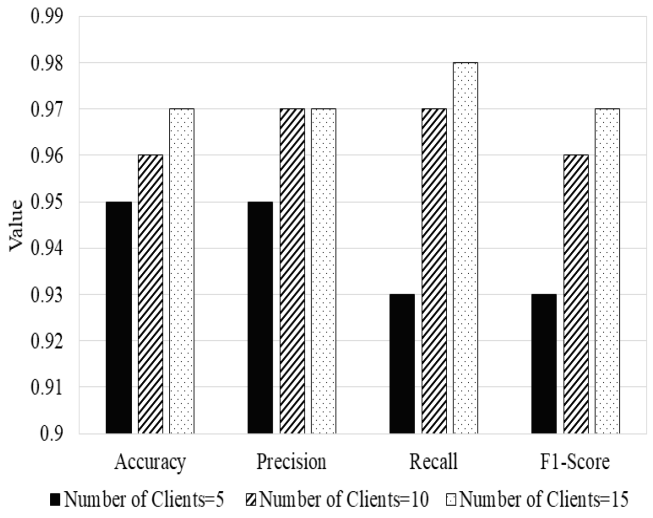

FIGURE 7 | Accuracy, precision, recall, and F1-score of global model in LSTM-based CFL.

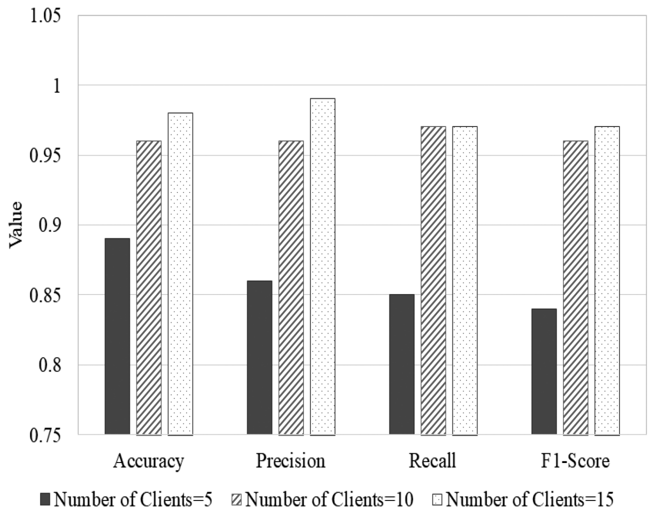

FIGURE 9 | Accuracy, precision, recall, and F1-score of global model in GRU-based CFL.

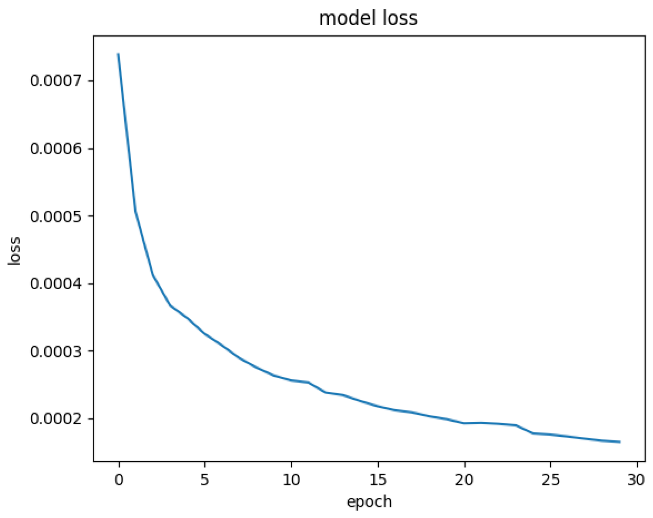

FIGURE 8 | Global model loss in LSTM-based CFL.

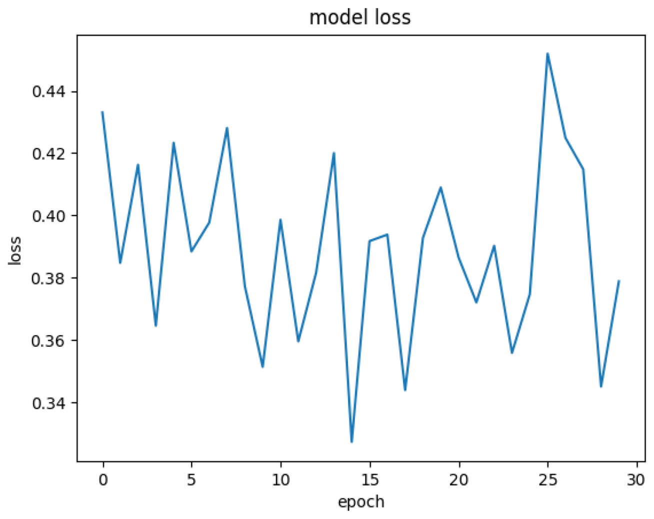

FIGURE 10 | Global model loss in GRU-based CFL.

LSTM has achieved an accuracy of 0.95 and 0.96, respectively. The global model loss using LSTM for Scenario 3 is presented in Figure 8. After ten rounds, the global loss is less than 0.0007 for the LSTM-based CFL framework.

The prediction accuracy, precision, recall, and F1-score of the global model using GRU for the considered three scenarios are presented in Figure 9. The global model using GRU has achieved a prediction accuracy of 0.98 in scenario 3. In scenarios 1 and 2, the global model using GRU has achieved an accuracy of 0.89 and 0.96, respectively. The global model loss using GRU for Scenario 3 is presented in Figure 10. As we observe for the GRU-based framework, the loss is below 0.44 after ten rounds.

As the number of clients participating in the CFL varies and each client has a different dataset, the prediction accuracy, precision, recall, and F1-score differ for different numbers of clients. We observe that the accuracy of the global model is higher (>0.96) for scenario 3 in both LSTM and GRU-based frameworks. The high accuracy, precision, recall, and F1-Score indicate the model provides accurate predictions with stability.

4.3.2 | Training Time

The average training time (per round) for the global model for scenarios 1, 2, and 3 using LSTM and GRU is presented in Figures 11 and 12, respectively. The training time is measured in seconds (s). As observed from the results, the average training times of the global model for scenarios 1, 2, and 3 while using LSTM are 64.13 s, 153.04 s, and 192.33 s, respectively. The average training times of the global model for scenarios 1, 2, and 3 while using GRU are 43.11 s, 124.52 s, and 165.23 s, respectively. Here, the training time is measured as the sum of the time consumption in model initialization, local model training, exchanging model updates with participating nodes (communication overhead), and aggregation, and then the average value is calculated based on the number of rounds. As the aggregation takes place only after receiving updates from all the participating clients, the

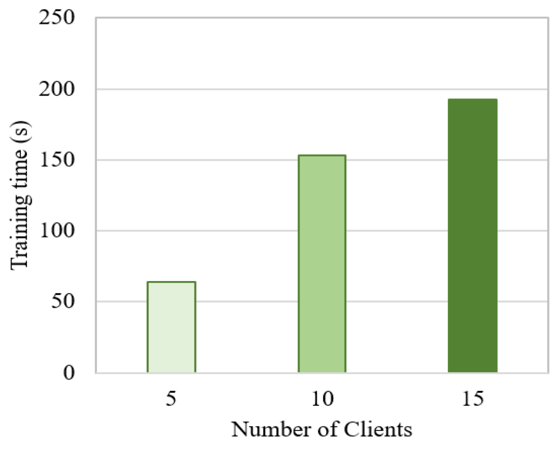

FIGURE 11 | Average training time (including communication) per round in CFL while using LSTM.

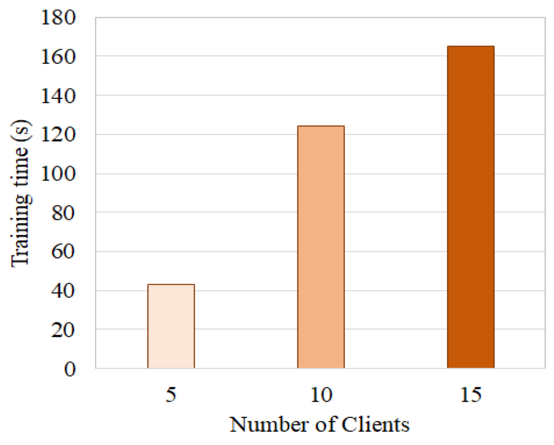

FIGURE 12 | Average training time (including communication) per round in CFL while using GRU.

training time increases with the number of participating clients. Thus, the training time for scenario 1 with five clients is low compared to the other two scenarios.

4.4 | DFL in Crop Yield Prediction

In this work, we used DFL frameworks with ring and mesh topology. Here, also, we considered three network scenarios: (i) Scenario 1: Four nodes form the network, (ii) Scenario 2: Seven nodes form the network, and (iii) Scenario 3: Ten nodes form the network. The maximum number of rounds we had considered was 10. In *Scenario 1*, we had considered four nodes (H1, H2, H4, and H5), which were connected either using ring topology or mesh topology. H3 worked as the server in our experiment. In the mesh-based network, each node shares its model updates with the rest of the nodes. In the considered ring-based P2P network, H1 shares its model updates with H2 and H5, H2 shares its model updates with H1 and H4, H4 shares its model updates, with H2 and H5, and H5 shares its model updates with H1 and H4. After receiving model updates each node performs aggregation and updates its local model accordingly.

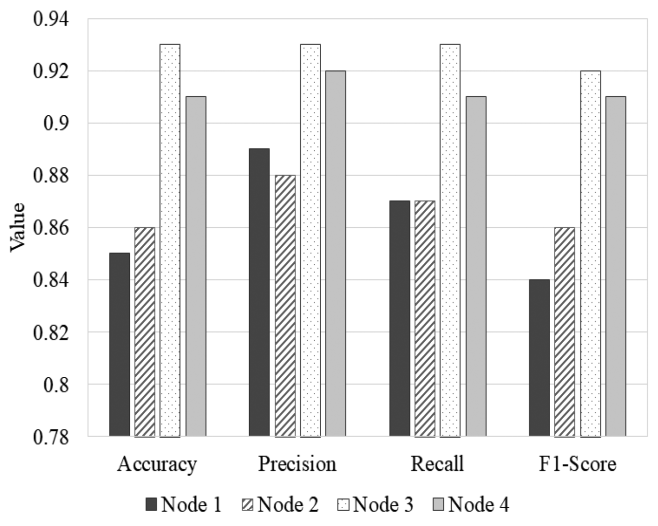

FIGURE 13 | Accuracy, precision, recall, and F1-score of the models using LSTM in ring-based DFL with four nodes.

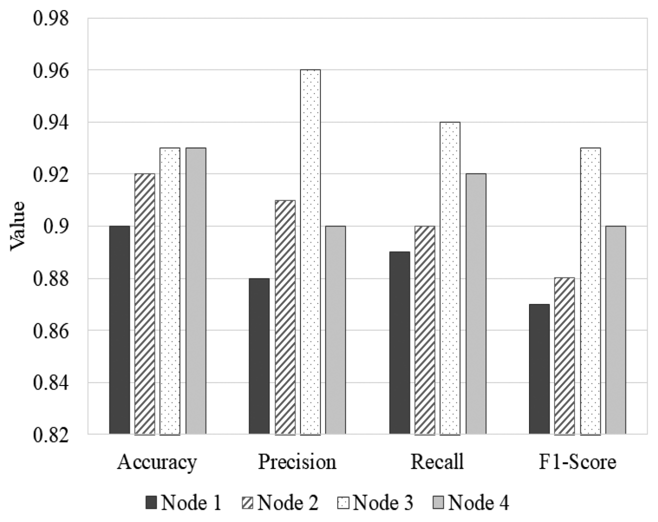

FIGURE 14 | Accuracy, precision, recall, and F1-score of the models using LSTM in mesh-based DFL with four nodes.

4.4.1 | Accuracy, Precision, Recall, and F1-Score

While using ring-based DFL with LSTM, the prediction accuracy, precision, recall, and F1-score of all four nodes are presented in Figure 13. The prediction accuracy, precision, recall, and F1-score of all four nodes while using mesh topology are presented in Figure 14.

We conducted the experiment for seven and ten nodes also. The prediction accuracy, precision, recall, and F1-score for the ring and mesh-based DFL frameworks using LSTM for all three scenarios are presented in Figures 15 and 16 respectively. We observe that for both the mesh and ring-based DFL frameworks, the accuracy for scenario 3 (number of nodes: 10) is higher. For scenario 3, the prediction accuracy is above 0.95 for both ring and mesh-based frameworks. The precision, recall, and F1-score are also above 0.95 for scenario 3. The global model loss for LSTM-based DFL frameworks for scenario 3 using ring and mesh

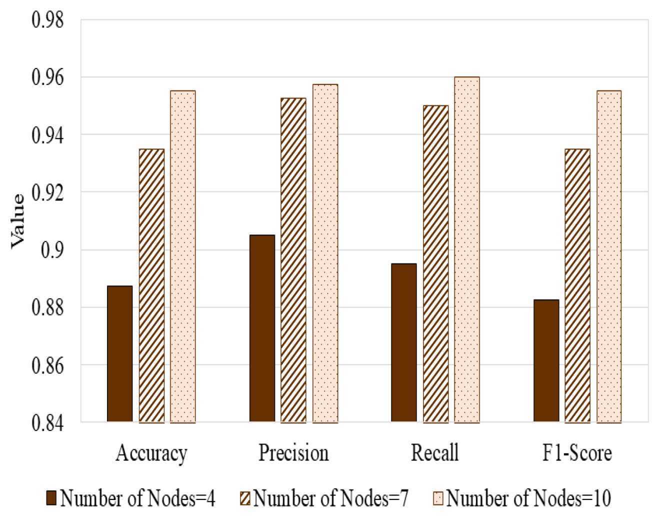

FIGURE 15 | Accuracy, precision, recall, and F1-score of the models using LSTM in ring-based DFL for three scenarios.

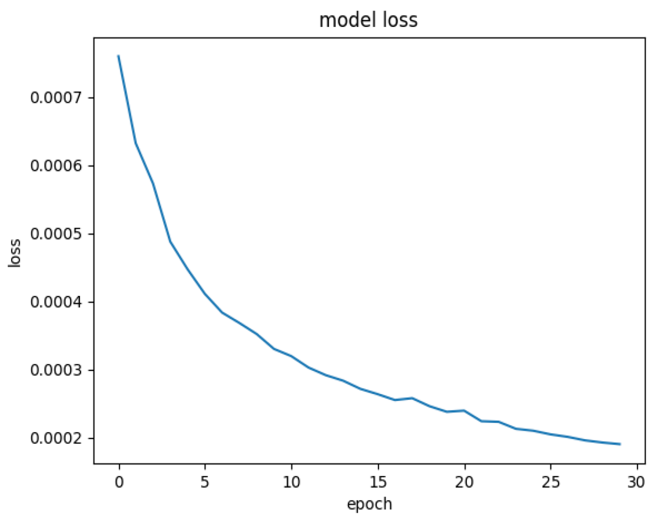

FIGURE 17 | Global model loss in LSTM-based DFL framework with ten nodes connected using ring topology.

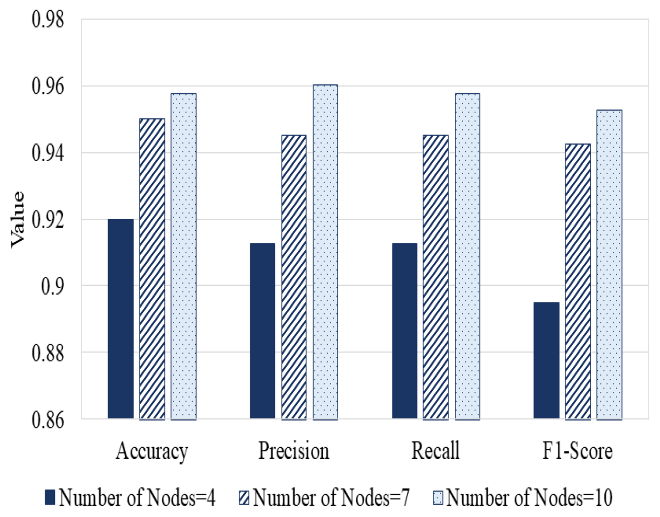

FIGURE 16 | Accuracy, precision, recall, and F1-score of the models using LSTM in mesh-based DFL for three scenarios.

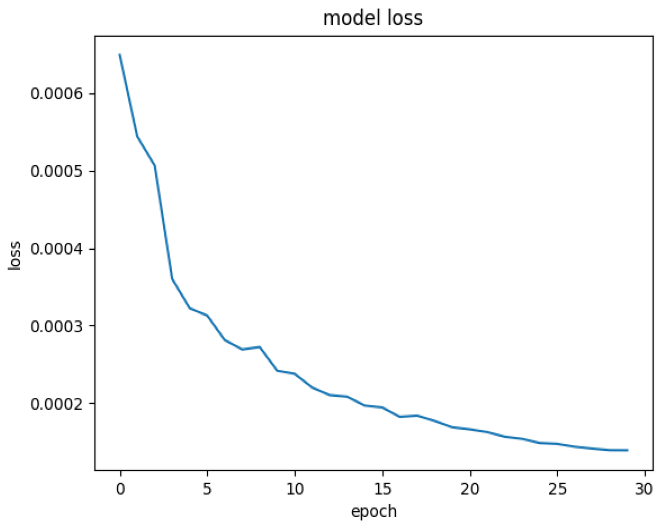

FIGURE 18 | Global model loss in LSTM-based DFL framework with ten nodes connected using mesh topology.

topology is presented in Figures 17 and 18 respectively. As we observe, the global model loss for the ring and mesh-based DFL frameworks is below 0.0007 and 0.0006, respectively, after ten rounds.

We have again conducted an experiment for scenario 3 where we considered GRU as the underlying model. As we observe using GRU, we obtained an accuracy, precision, recall, and F1-Score of 0.935, 0.9325, 0.945, and 0.9275, respectively, for the ring-based DFL framework. The accuracy, precision, recall, and F1-score for the mesh-based DFL framework using GRU are 0.955, 0.9525, 0.96, and 0.9525, respectively. The results are pictorially presented in Figure 19. The global model loss for GRU-based DFL frameworks for scenario 3 using ring and mesh topology is presented in Figures 20 and 21 respectively. We observe that after ten rounds, the global model losses are below 0.46 for both the ring and mesh-based DFL frameworks using GRU.

4.4.2 | Training Time

The average training time (per round) of the four nodes in ring and mesh-based frameworks for scenario 1 is presented in Figure 22. As we observed in the ring-based framework, the average training times of nodes 1, 2, 3, and 4 are 35.53 s, 27.86 s, 31.84 s, and 30.62 s, respectively. We also observed that the average training times of the nodes 1, 2, 3, and 4 are 36.99 s, 39.65 s, 36.8 s, and 32.97 s, while using the mesh-based framework. We have measured the training time for scenarios 2 and 3 also. The average training times (per round) for all three scenarios are presented in Figure 23. Here, we have used LSTM as the underlying model.

As observed from the results, the average training times (per round) of the local models for scenarios 1, 2, and 3 are 31.46 s, 32.26 s, and 30.98 s, respectively, while using ring topology. We also observed that while using mesh topology, the average

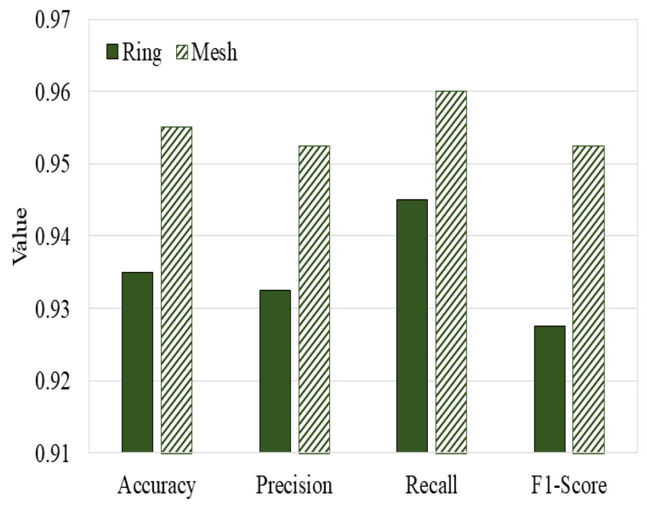

FIGURE 19 | Accuracy, precision, recall, and F1-score of the models in the GRU-based DFL framework with ten nodes.

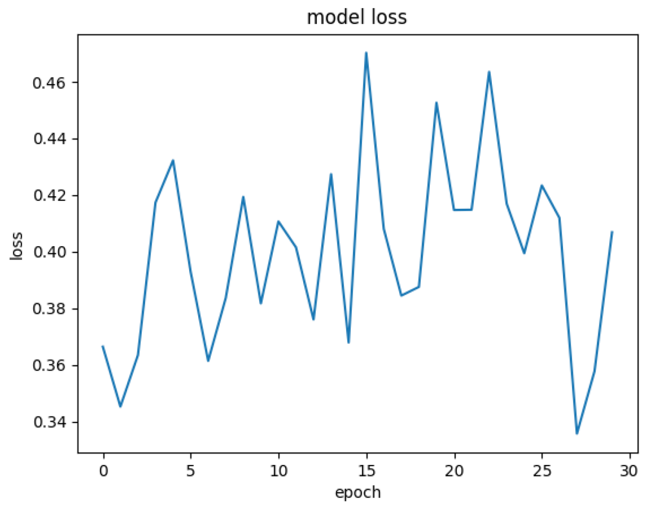

FIGURE 20 | Global model loss in GRU-based DFL framework with ten nodes connected using ring topology.

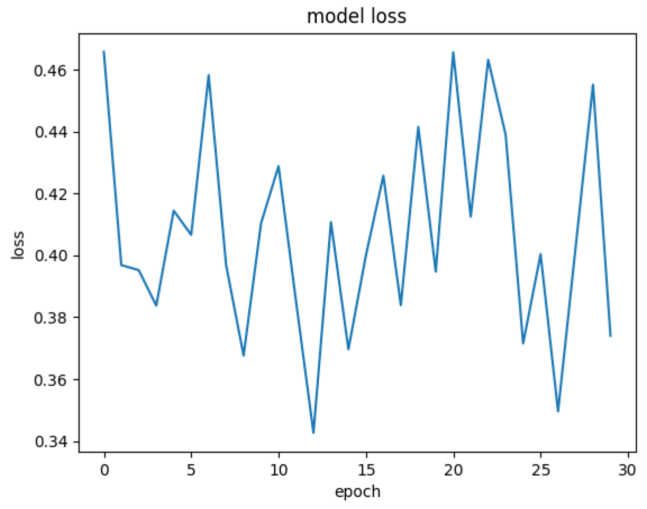

FIGURE 21 | Global model loss in GRU-based DFL framework with ten nodes connected using mesh topology.

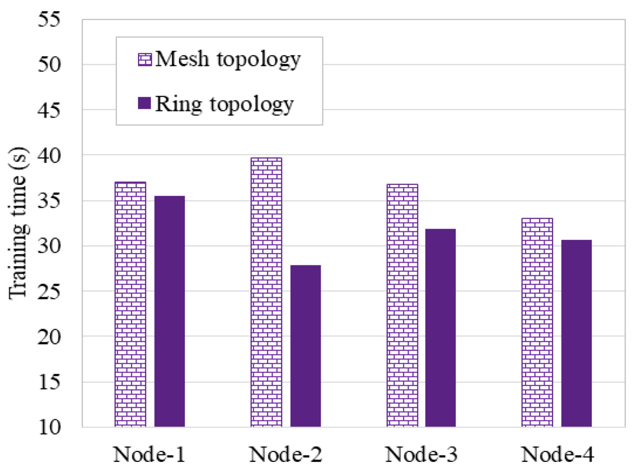

FIGURE 22 | Average Training time (including communication) of the nodes in ring and mesh-based frameworks.

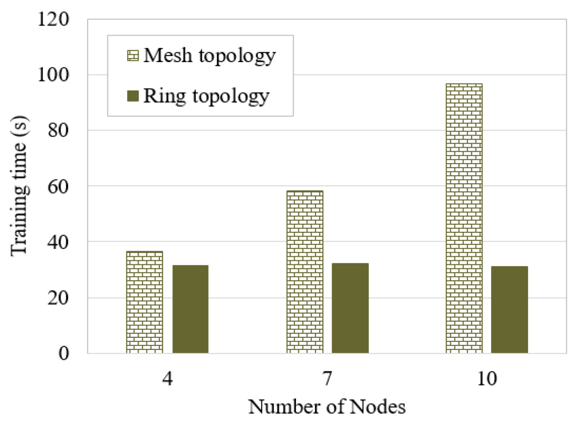

FIGURE 23 | Average training time (including communication) per round in ring and mesh-based frameworks.

training times (per round) of the local models for scenarios 1, 2, and 3 are 36.6 s, 58.4 s, and 96.71 s, respectively. In ring topology, each node exchanges model updates with the adjacent nodes, whereas in mesh topology, each node exchanges model updates with the rest of the nodes in the formed network. As we observed in our experiment, the training time for mesh topology was higher. We also measured the response time for both the mesh-based and ring-based frameworks, and we observed that for both ring and mesh topology, the response time was in the range of 1.2–3.5 s, in all three cases. The average training times per round for the ring and mesh-based frameworks with ten nodes are 20.35 s and 85.53 s, respectively, while using GRU as the underlying model. The response time for the ring and mesh-based DFL frameworks using GRU was approximately 3.2 s, for scenario 3.

Finally, we observe that (i) for CFL with fifteen nodes, we achieved >96% accuracy, and (ii) for DFL using ring and mesh topology with ten nodes, we achieved >93% prediction accuracy. As we observe from the experimental results, using CFL and DFL, high prediction accuracy can be obtained without sharing the data.

4.5 | Comparison With Cloud-Only Model

In Figures 24 and 25, the accuracy, precision, recall, and F1-Score of the global model for the CFL framework using LSTM and GRU are compared to the cloud-only framework, where the entire dataset is analyzed inside the cloud. The number of participating nodes was 15. The results show that the use of collaborative learning using CFL has improved the accuracy, precision, recall, and F1-score by ∼5%–8% than the cloud-only framework for both LSTM and GRU. Figures 26 and 27 present the comparison of the response time of the CFL-based model to the cloud-only framework using LSTM and GRU, respectively. The results show that the CFL-based framework reduces the response time ∼75% than the cloud-only framework.

4.6 | Comparison With Existing Approaches

In this section, we compare the performance of the CFL and DFL-based frameworks in crop yield prediction with the state-of-the-art models for crop yield prediction. The comparative analysis is presented in Table 4. At first, we compare the CFL and DFL-based frameworks with the existing crop yield prediction frameworks that used the *same dataset*[1] that we used for performance analysis. After that we draw a comparison of the CFL and DFL-based frameworks with an existing FL-based framework that used another dataset for performance evaluation.

In [12], the authors used RF, DT, KNN, XGBoost, and SVM, which are well-known ML models, and among them RF achieved the highest accuracy of 97.18%. In [13], MLP, decision table, and JRip were used, and MLP achieved the highest accuracy of 98.23% in crop yield prediction. KNN was adopted in [14] for data analysis, and a recall value of 92.62% was achieved for crop yield prediction. In [15], the authors used DT, SVM, KNN, LGBM, and RF, in crop yield prediction, and among them RF achieved the highest accuracy of 99.24%. The authors in [16] used an LSTM, Bi-LSTM, and GRU-based framework for data analysis and achieved an accuracy of 98.45% in crop yield prediction. In [18], Bi-LSTM was used for crop yield prediction, and the achieved accuracy was 98.64%. As we observe from Table 4, none of the existing approaches used FL. Most of the existing approaches focused on the use of ML/DL approaches for crop yield prediction without addressing the concern of using a cloud-only paradigm for data analysis, such as network connectivity issues, data privacy, response time, etc. To achieve data privacy protection without

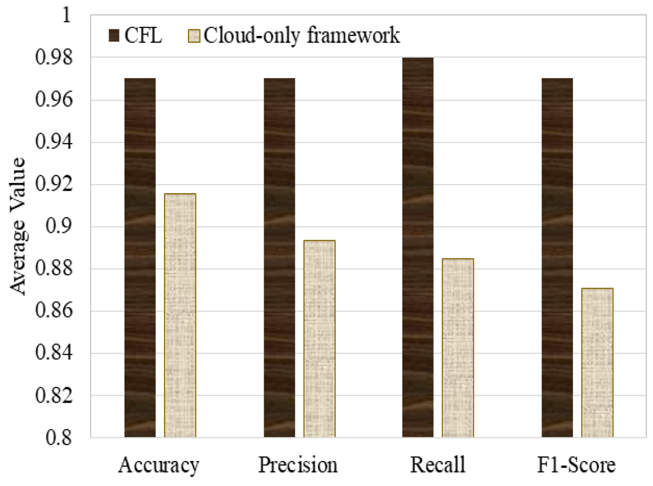

FIGURE 24 | Comparison of accuracy, precision, recall, and F1-score between the CFL-based and Cloud-only frameworks using LSTM.

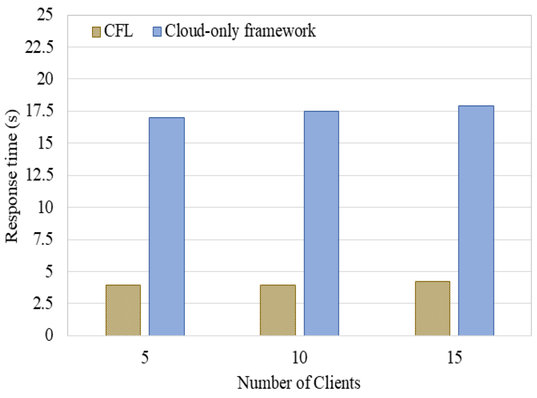

FIGURE 26 | Comparison of response time between the CFL-based and Cloud-only frameworks using LSTM.

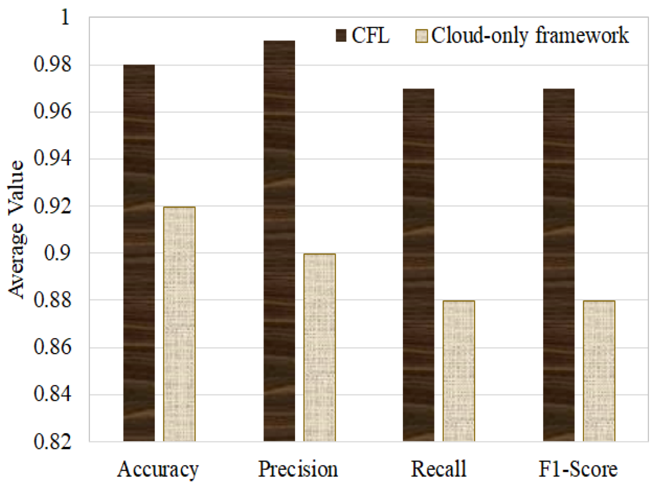

FIGURE 25 | Comparison of accuracy, precision, recall, and F1-score between the CFL-based and Cloud-only frameworks using GRU.

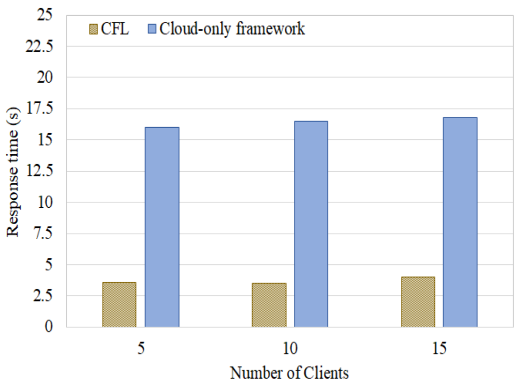

FIGURE 27 | Comparison of response time between the CFL-based and Cloud-only frameworks using GRU.

TABLE 4 | Comparison of performance of proposed and existing crop yield prediction frameworks.

Work	Classifier	FL is used	Accuracy	Training time per round	Response time
[12]	RF (Highest) DT, KNN XGBoost, SVM	No	97.18%	Not measured	Not measured
[13]	MLP (Highest) Decision table, JRip	No	98.23%	Model build time: 10.56 s (MLP)	Not measured
[14]	KNN	No	92.62% (Recall)	Not measured	Not measured
[15]	DT, SVM KNN, LGBM RF (Highest)	No	99.24%	Not measured	Not measured
[16]	LSTM, Bi-LSTM, GRU	No	98.45%	Not measured	Not measured
[18]	Bi-LSTM	No	98.64%	Model build time: 0.53 s	Latency: 4.54–4.91 s
[25]	Gaussian NB	Yes (CFL)	90%	Not measured	Not measured
FL-based framework	LSTM, GRU	Yes (CFL and DFL)	CFL: 97% (LSTM) 98% (GRU) (15 clients) DFL: >95% (LSTM) >93% (GRU) (Ring) >95% (LSTM) >95% (GRU) (Mesh) (10 nodes)	CFL: 192.33 s (LSTM) 165.23 s (GRU) (15 clients) DFL: 30.98 s (LSTM) 20.35 s (GRU) (Ring) 96.71 s (LSTM) 85.53 s (GRU) (Mesh) (10 nodes)	CFL: 2.5–5 s (5–15 clients) DFL: 1.2–3.5 s (4–10 nodes)

sharing data but obtain a model with high prediction accuracy through collaborative training has been addressed in our work. We have explored the use of FL in crop yield prediction through an experimental analysis using multiple clients. As we observe, CFL and DFL have achieved high prediction accuracy but without sharing actual datasets. Hence, we observe that using FL, high prediction accuracy can be obtained like the state-of-the-art models but with enhanced data privacy. In [25], CFL was used for crop yield prediction based on Gaussian NB, and 90% prediction accuracy was achieved with Adam optimizer and learning rate 0.001. As we observe, only CFL was used in [25], whereas we have used both CFL and DFL in crop yield prediction based on LSTM and GRU. Further, we have achieved higher accuracy (>96%) than [25] (Optimizer: Adam, learning rate: 0.001), while using CFL. Further, the training time for both the CFL and DFL approaches has been determined in our work, and we observe that the training time is medium for the considered scenarios. We also observe that the response time is low for both the CFL and DFL-based frameworks. Thus, crop yield prediction with high accuracy but low response time can be achieved using FL-based frameworks.

5 | Another Case Study: Road Traffic Monitoring

To evaluate the performance of the proposed FL architectures, we considered another application also. As the underlying model, GRU was used. In this case, we considered the dataset of road traffic monitoring[3]. From the dataset, the input parameters considered were the number of cars, bikes, buses, and trucks, and the output is the traffic situation (low, normal, high, heavy). There are 5952 samples in the dataset. The number of rounds considered in both CFL and DFL was 10.

For implementing the CFL scenario, we used three VM instances as the clients and one VM instance as the server. The dataset was divided into four parts. Three parts were assigned to the clients, and one part was assigned to the server. The accuracy, precision, recall, and F1-score for the global and local models are presented in Figure 28.

To implement the DFL scenario, we considered four VMs that were connected using mesh topology. The dataset is divided into four parts and assigned to these four nodes. The accuracy, precision, recall, and F1-score for the four nodes in the DFL scenario are presented in Figure 29. The average training time, including communication for both CFL and DFL scenarios, was approximately 45 s. We observe from the second case study that the CFL-based framework has a prediction accuracy of 0.96 for the global model and ≥0.95 for the local models. The precision, recall, and F1-score are also 0.96 for the global model and ≥0.95 for the local models in the CFL-based framework. The prediction accuracy, precision, recall, and F1-score are >0.95 for the DFL-based framework. The global model loss for both the CFL and DFL

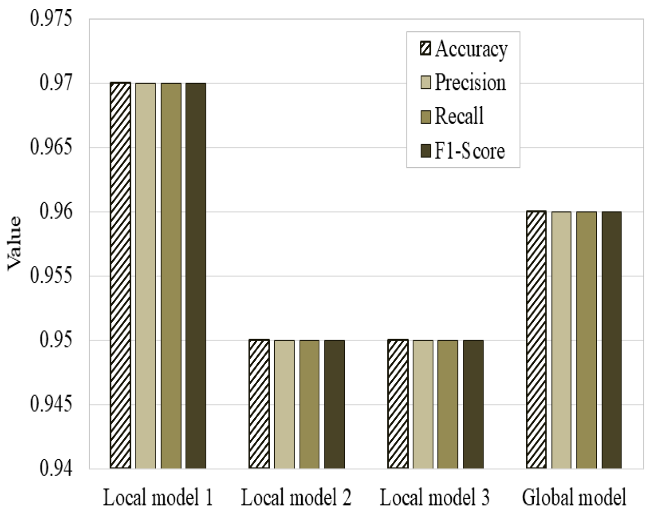

FIGURE 28 | Accuracy, precision, recall, and F1-score of the models in CFL for case study 2.

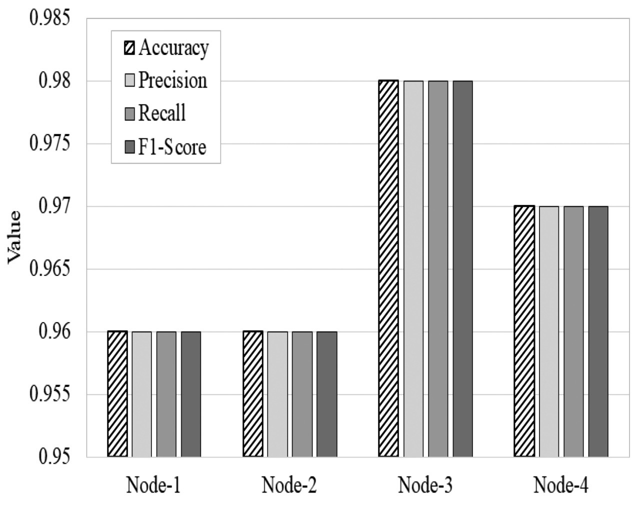

FIGURE 29 | Accuracy, precision, recall, and F1-score of the models in DFL for case study 2.

scenarios is less than 0.15 after ten rounds. The low loss and high prediction accuracy, precision, recall, and F1 score denote that both the CFL and DFL frameworks are stable and can accurately predict the road traffic.

For both the case studies (crop yield prediction and road traffic prediction), the CFL and DFL architectures show good performance in terms of prediction accuracy and loss. The training time, including communication for both case studies, is also not very high, as observed from the results. Further, as no raw data is shared in FL, the data privacy is protected.

6 | Future Research Directions

In this work, we have highlighted the use of CFL and DFL in crop yield prediction. However, there still remain several open challenges discussed below:

- Data heterogeneity: Data heterogeneity is a critical issue of FL. As the data is distributed among several clients, it may lead to non-independent, identically distributed, and unbalanced datasets. In such a scenario, model training is a challenge, and it becomes critical to build a global model with consistent performance across all the clients.
- Use of FTL: The datasets of different clients may have different sample spaces as well as different feature spaces. In that case, FTL can be used. In FTL, features from different feature spaces are transferred to the same presentation. Further, for enhancing data privacy and security, gradient updates are encrypted. The use of FTL with gradient encryption is a significant research direction.
- Resource limitation of user device: The FL encourages local data analysis and collaborative learning. However, the user device may not have sufficient resources for executing an ML/DL algorithm, and the user has to use the cloud server for data analysis. Another difficulty may arise when a device cannot execute its local model due to resource limitation or any other issue. In that case, the model of that device along with the dataset needs to be transferred to a nearby node. In both scenarios, cryptography or steganography can be used to protecting the data privacy by either encrypting or hiding it inside a media during transmission.
- Enhance the security of model parameters and the system: Though no data is shared, and only the model updates are exchanged in FL, there is still a possibility of leakage of gradient information. In such a scenario, gradient encryption can be used. Further, blockchain [26] can be integrated with FL for enhancing the security of the entire system.
- Communication overhead: In FL, the exchange of model updates during training enhances the communication overhead and latency. Therefore, a trade-off should be maintained between the number of rounds of training the model and communication overhead so that prediction accuracy can be good but the latency will not be very high.
- Straggler effect: The straggler effect is another challenge of FL, where the training inside the local devices may slow down the devices that degrade the overall performance. Federated offloading can be used as a solution to this issue [27]. In federated offloading, one part of the dataset is used for local training, and the other part is offloaded to a nearby lightly loaded device or to the cloud. The offloaded data is used for training inside the lightly loaded device or inside the cloud. After each round, the parameter weights from both the training are used for aggregation. Cryptography can be used for protecting data during offloading.

7 | Conclusions

Crop yield prediction is a crucial area of smart agriculture. In this paper, we have explored the use of CFL and DFL in crop yield prediction based on LSTM and GRU. An experimental case study has been conducted, where different numbers of devices perform collaborative training using CFL and DFL. To implement CFL, a client-server paradigm is developed using MLSocket, and multiple clients are handled by the server. To implement

DFL, a collaborative network is formed among the nodes using ring topology and mesh topology. In the ring-based P2P framework, each node exchanges model updates with the neighbor nodes and performs aggregation to build the upgraded model. In the mesh-based framework, each node exchanges model updates with rest of the nodes and performs aggregation to build the upgraded model. The performance of the CFL and DFL-based frameworks is evaluated in terms of prediction accuracy, precision, recall, F1-Score, and training time. If the communication with the cloud server is not good, DFL can be adopted. The DFL framework with mesh topology can have more communication overhead compared to the ring-based framework if the number of participating nodes is high. However, good prediction accuracy is achieved by both the CFL and DFL frameworks without compromising with data privacy. The experimental results show that >93% prediction accuracy has been achieved using the CFL and DFL-based frameworks. The results also show that the CFL-based framework reduces the response time ~75% than the cloud-only framework. The average accuracy, precision, recall, and F1-Score are also improved by ~5%–8% using CFL than the cloud-only framework. As we observe from the experimental implementations of the CFL and DFL architectures, for both cases high prediction accuracy with low loss is achieved for the considered applications. Finally, the future research directions in crop yield prediction are highlighted in this paper.

Author Contributions

The author takes full responsibility for this article.

Acknowledgments

This work is partially supported by an ARC Discovery Project (DP240102088) and ARC LIEF. Open access publishing facilitated by The University of Melbourne, as part of the Wiley - The University of Melbourne agreement via the Council of Australian University Librarians.

Data Availability Statement

Data sharing is not applicable to this article as no new data were created or analyzed in this study.

Endnotes

[1] https://github.com/AnuTuli/FL-Implementation.

[2] https://www.kaggle.com/datasets/atharvaingle/crop-recommendation-dataset.

[3] https://www.kaggle.com/datasets/hasibullahaman/traffic-prediction-dataset?select=TrafficTwoMonth.csv.

References

1. A. D. Boursianis, M. S. Papadopoulou, P. Diamantoulakis, et al., "Internet of Things (IoT) and Agricultural Unmanned Aerial Vehicles (Uavs) in Smart Farming: A Comprehensive Review. Internet of Things," *Internet of Things* 18 (2022): 100187.

2. O. Debauche, J. P. Trani, S. Mahmoudi, et al., "Data Management and Internet of Things: A Methodological Review in Smart Farming," *Internet of Things* 14 (2021): 100378.

3. S. Bera, T. Dey, S. Ghosh, and A. Mukherjee, "Internet of Things and Dew Computing-Based System for Smart Agriculture," in *Dew Computing: The Sustainable IoT Perspectives* (Springer, 2023), 289–316.

4. S. Bera, T. Dey, A. Mukherjee, and R. Buyya, "E-Cropreco: A Dew-Edge-Based Multi-Parametric Crop Recommendation Framework for Internet of Agricultural Things," *Journal of Supercomputing* 79, no. 11 (2023): 11965–11999.

5. S. Bera, T. Dey, A. Mukherjee, and D. De, "Flag: Federated Learning for Sustainable Irrigation in Agriculture 5.0," *IEEE Transactions on Consumer Electronics* 70, no. 1 (2024): 2303–2310.

6. L. Li, Y. Fan, M. Tse, and K. Y. Lin, "A Review of Applications in Federated Learning," *Computers & Industrial Engineering* 149 (2020): 106854.

7. D. C. Nguyen, M. Ding, P. N. Pathirana, A. Seneviratne, J. Li, and H. V. Poor, "Federated Learning for Internet of Things: A Comprehensive Survey," *IEEE Communications Surveys & Tutorials* 23, no. 3 (2021): 1622–1658.

8. C. Zhang, Y. Xie, H. Bai, B. Yu, W. Li, and Y. Gao, "A Survey on Federated Learning," *Knowledge-Based Systems* 216 (2021): 106775.

9. V. Mothukuri, R. M. Parizi, S. Pouriyeh, Y. Huang, A. Dehghantanha, and G. Srivastava, "A Survey on Security and Privacy of Federated Learning," *Future Generation Computer Systems* 115 (2021): 619–640.

10. Y. Djenouri, T. P. Michalak, and J. C. W. Lin, "Federated Deep Learning for Smart City Edge-Based Applications," *Future Generation Computer Systems* 147 (2023): 350–359.

11. T. Van Klompenburg, A. Kassahun, and C. Catal, "Crop Yield Prediction Using Machine Learning: A Systematic Literature Review," *Computers and Electronics in Agriculture* 177 (2020): 105709.

12. N. N. Thilakarathne, M. S. A. Bakar, P. E. Abas, and H. Yassin, "A Cloud Enabled Crop Recommendation Platform for Machine Learning-Driven Precision Farming," *Sensors* 22, no. 16 (2022): 6299.

13. K. Bakthavatchalam, B. Karthik, V. Thiruvengadam, et al., "Iot Framework for Measurement and Precision Agriculture: Predicting the Crop Using Machine Learning Algorithms," *Technologies* 10, no. 1 (2022): 13.

14. M. Cruz, S. Mafra, and E. Teixeira, "An IoT Crop Recommendation System With K-NN and Lora for Precision Farming," 2022.

15. P. Kathiria, U. Patel, S. Madhwani, and C. Mansuri, "Smart Crop Recommendation System: A Machine Learning Approach for Precision Agriculture," in *Machine Intelligence Techniques for Data Analysis and Signal Processing: Proceedings of the 4th International Conference MISP 2022*, vol. 1 (Springer, 2023), 841–850.

16. P. Gopi and M. Karthikeyan, "Red Fox Optimization With Ensemble Recurrent Neural Network for Crop Recommendation and Yield Prediction Model," *Multimedia Tools and Applications* 83, no. 5 (2024): 13159–13179.

17. P. M. Gopal and R. Bhargavi, "A Novel Approach for Efficient Crop Yield Prediction," *Computers and Electronics in Agriculture* 165 (2019): 104968.

18. T. Dey, S. Bera, B. Paul, D. De, A. Mukherjee, and R. Buyya, "Fly: Femtolet-Based Edge-Cloud Framework for Crop Yield Prediction Using Bidirectional Long Short-Term Memory," *Software: Practice and Experience* 54, no. 8 (2024): 1361–1377.

19. W. Zhu, M. Goudarzi, and R. Buyya, "Flight: A Lightweight Federated Learning Framework in Edge and Fog Computing," *Software: Practice and Experience* 54, no. 5 (2024): 813–841.

20. S. B. Atitallah, M. Driss, and H. B. Ghezala, "Fedmicro-ida: A Federated Learning and Microservices-Based Framework for IoT Data Analytics," *Internet of Things* 23 (2023): 100845.

21. T. Manoj, K. Makkithaya, and V. Narendra, "A Federated Learning-Based Crop Yield Prediction for Agricultural Production Risk Management," in *Proceedings of the 2022 IEEE Delhi Section Conference (DELCON)* (IEEE, 2022), 1–7.

22. O. Friha, M. A. Ferrag, L. Shu, L. Maglaras, K. K. R. Choo, and M. Nafaa, "Felids: Federated Learning-Based Intrusion Detection System

for Agricultural Internet of Things," *Journal of Parallel and Distributed Computing* 165 (2022): 17–31.

23. A. Li, M. Markovic, P. Edwards, and G. Leontidis, "Model Pruning Enables Localized and Efficient Federated Learning for Yield Forecasting and Data Sharing," *Expert Systems with Applications* 242 (2024): 122847.

24. A. Durrant, M. Markovic, D. Matthews, D. May, J. Enright, and G. Leontidis, "The Role of Cross-silo Federated Learning in Facilitating Data Sharing in the Agri-Food Sector," *Computers and Electronics in Agriculture* 193 (2022): 106648.

25. G. Idoje, T. Dagiuklas, and M. Iqbal, "Federated Learning: Crop Classification in a Smart Farm Decentralised Network," *Smart Agricultural Technology* 5 (2023): 100277.

26. S. Bera, T. Dey, A. Mukherjee, P. Bhattacharya, and D. De, "Fedchain: Decentralized Federated Learning and Blockchain-Assisted System for Sustainable Irrigation," *IEEE Transactions on Consumer Electronics* (2024).

27. A. Mukherjee and R. Buyya, "A Joint Time and Energy-Efficient Federated Learning-Based Computation Offloading Method for Mobile Edge Computing," 2024 arXiv preprint arXiv:2409.02548.