FLyer: Federated Learning-based Crop Yield Prediction for Agriculture 5.0

***Abstract*—Crop yield prediction is a significant area of precision agriculture. In this paper, we propose a crop yield prediction framework named FLyer, based on federated learning and edge computing. In FLyer, the soil and environmental data are locally processed inside the edge servers, and the model parameters are transmitted between the edge servers and the cloud with encrypted gradients. LSTM is used as the local and global models for data analysis. As the LSTM model can capture the temporal dependencies and hold the sequential nature of the data, we use LSTM in FLyer. By encrypting the gradients, the gradient information leakage ratio is reduced, and data privacy is protected. For gradient encryption, we use AES-256, and for data encryption during local storage we use RSA and AES-256. The results demonstrate that FLyer diminishes the latency by ∼39% and energy consumption by ∼40% than the conventional edge-cloud framework respectively. The experimental results show that the global model in FLyer achieves above 99% accuracy, precision, recall, and F1-score in crop yield prediction. The results also present that the local models also achieve >94% accuracy in crop yield prediction.**

***Impact Statement*—The proposed framework FLyer provides crop yield prediction with high accuracy and data privacy protection. The proposed approach is based on federated learning. The soil and environmental data are processed locally inside the edge servers using LSTM. The model parameters are exchanged between the edge servers and the cloud after encrypting gradients. The use of encryption during model parameters' exchange has reduced gradient information leakage ratio to 0.038 for 50% malicious participants. From the results we observe that FLyer has achieved above 99% accuracy in crop yield prediction. We also have observed that FLyer has reduced the latency by ∼39% and energy consumption by ∼40% than the edge-cloud framework without federated learning respectively.**

***Index Terms*—Federated learning, Latency, Energy, Crop yield prediction, Gradient encryption.**

I. Introduction

THE transition regarding sustainability in Agriculture 5.0 [1] represents a significant transformation, where technological advancements align with environment, society, and economy. Artificial Intelligence (AI) is incorporated with the Internet of Things (IoT) for developing smart systems. Agriculture 5.0 emphasizes a comprehensive strategy that goes beyond the traditional methods in the field of smart agriculture. The convergence of advanced technology, such as AI and data analytics, signifies a new era when precision farming aligns with environmental awareness. The IoT-based crop recommendation system is a pioneering agricultural solution that represents a remarkable technological achievement in the era of smart farming. The integration of sensors, data analytics, and machine learning, provides farmers with suggestions customized to the respective soil and environmental parameters' values. By collecting and analyzing data on soil and environmental factors such as soil moisture, soil nutrients, temperature, and humidity, the system provides farmers with insightful guidance regarding planting, irrigation, and fertilization.

In the realm of agricultural data analytics, cloud computing performs as the backbone for processing huge amounts of data generated using IoT devices [2]. Nevertheless, the agricultural lands are frequently located across rural areas, where Internet connectivity is unstable. As a result, the transfer of data to cloud servers experiences disruptions, which leads to increased delay in data transmission. Edge computing offers a strategic solution by placing computational resources at the network edge [2], [3], [4]. Furthermore, cache-based dew computing [2], [5] enables the temporary storage of data when Internet connectivity is unavailable. In dew computing, the dew servers provide micro-services, and a dynamic framework is established with the cloud for periodic data synchronization. The concern over delays, data traffic, and cloud overhead in the conventional model for transmitting all data for storage and processing in the cloud, prompts the integration of Federated Learning (FL). In FL, multiple clients perform local data analysis with their datasets using the model initiated by the central server and send the model parameters to the server connected to the clients [6]. The server revises the global model and then sends the revised model parameters to the clients. In an FL-based edge-cloud model, the clients are the local edge nodes, which communicate their updated weights to the cloud server. The cloud compiles those updates and sends the resulting combined model back to the edge nodes for further training. This process is repeated until the convergence is reached [7]. This strategy provides the local-level data processing facility, and the model parameters are exchanged between the clients and the server to update the global model accordingly [8], [9]. FL reduces data traffic significantly by locally processing and storing the data. FL employs the obtained weighted parameters to improve the training process, providing a more efficient and privacy-focused approach compared to the traditional way of transmitting data to the cloud for storage and processing purposes [10].

In the agricultural domain, variations in soil, environment, and weather parameters are inherent and depend upon geographical diversity. Moreover, each farm has unique soil compositions and specific weather requirements. Hence, the data of each farm is confidential. FL [11] is a crucial approach that permits to train model on devices or edge nodes, eliminating the requirement for centralized data exchange. The strategic implementation of FL not only maintains privacy by avoiding the sharing of raw data but also enables a flexible adjustment to individual circumstances [12]. The model is finely adjusted to explore the convergence of the decentralized models while optimizing the Federated Averaging model of the smart farming

dataset [7]. The implementation of FL-based systems promotes collaborative efforts among multiple users, as they collectively contribute to improve the prediction models while maintaining the secrecy of individual data. Hence, the use of FL in crop yield prediction may lead to a significant improvement in the overall performance of the system by enhancing data privacy and prediction accuracy of the system.

A. Motivations and Contributions

Latency, energy consumption, and sustainability are crucial factors for any digitized framework of next-generation networks. The objective is to construct a secure, low-latency, and energy-efficient framework that will predict crop yield. This paper makes the following key contributions:

- An *FL*-based crop *y*ield pr*e*diction f*r*amework referred to as *FLyer* is proposed. The rural regions mainly contain agricultural lands, where network connectivity is not good. The dew computing based on cache, offers the provision of temporary data storage inside the user device, when Internet connection is not available.
- The FL along with edge computing is used to provide local data processing facility that helps to reduce the latency, and the privacy is preserved by avoiding raw data sharing. Also, during model parameters' exchange between the edge nodes and cloud, the gradients are encrypted to prevent gradient information leakage. The local data storage inside the edge servers takes place in encrypted form for data security purpose.
- For data analysis, Long Short-Term Memory (LSTM) network is used in the local and global models. The training on real data in the distributed edge nodes gives a benefit over training on centralized data inside the cloud from the perspective of latency and energy consumption.

B. Layout of the paper

The rest of the paper is organized as follows: Section II presents a brief description of the existing literature. Section III illustrates the proposed framework FLyer. Section IV analyses the performance of the FLyer. Section V presents conclusion with future research directions.

II. Related Work

Crop yield prediction is a vital area of smart agriculture. In [13], IoT-based architecture provided extensive data collection in agriculture, improving crop prediction models using machine learning (ML). The studies employing algorithms like Multilayer Perception (MLP), JRip, and decision tables demonstrated significant accuracy in forecasting optimal crops [13]. In [14], Multivariate Linear Regression, k-Nearest Neighbor (KNN), and Decision Tree (DT), were employed to forecast the yields of six different crops. For crop yield prediction, KNN was used in [15]. Various ML models such as Support Vector Machine (SVM), Light Gradient Boosting Machine (LGBM), Radom Forest (RF), KNN, and DT, were used for crop yield prediction in [16]. The deep learning models such as Gated Recurrent Unit (GRU), LSTM, and Bidirectional-LSTM (Bi-LSTM)-based crop yield prediction framework was proposed in [17]. In [6], [18], FL was used for irrigation decision-making, and the use of FL achieved high prediction accuracy. Various Deep Neural Networks (DNNs) were specifically developed to forecast soil moisture by utilizing data obtained from diverse sources in [19]. The incorporation of AI, particularly ML, revolutionized precision agriculture by enabling smarter, data-driven solutions for improving crop management [20]. The research concentrated on assessing algorithms such KNN, DT, RF, Extreme Gradient Boosting (XGBoost), and Support Vector Machine (SVM) to determine the most effective method for improving agricultural decision-making [20]. In [21], FL utilizing Residual Networks (ResNet)-based models demonstrated efficacy in optimizing yield forecasts inside decentralized environments. It exhibited enhanced performance relative to conventional centralized architectures [21]. A federated RF algorithm was introduced in [22] to allow breeding institutions to interact without revealing or exchanging their data. A study was conducted to empirically validate the joint modeling of locally stored tabular data, including field breeding trial phenotypes and climatic conditions, for predicting maize variety yield [22]. An approach of integrating data from several sensors was implemented in [23] to categorize eight different crops using DT, Hoeffding Tree, and RF. An alternative ML method was proposed in [24] to enhance the precision of forecasting Plant Nitrogen Concentration (PNC) during the critical growth stages of wheat, maize, rice, and potato. In [7], the implementation of FL was explored in smart farming. The Federated Averaging model was utilized to classify crops by using climate indicators as predictors and crop categories as output labels [7].

In assessing several studies on ML and FL applications in agriculture, it is crucial to evaluate both the pros and cons of each method. Idoje et al. [7] emphasized the benefits of privacy preservation, and focused on secure data management. However, the scope of the study was constrained by a limited dataset, and the latency and energy consumption were not considered [7]. Zhang et al. [22] implemented rigorous privacy safeguards, supported by empirical validation. However, the latency and energy consumption were not considered in [22]. Reyana et al. [23] focused on the integration of multisensor data to refine precision in agricultural forecasts, representing a notable advantage, particularly in enhancing accuracy. However, the study was confined to particular crops, a limited set of ML algorithms were utilized, and the latency and energy consumption were not considered [23]. Thilakarathne et al. [20] presented a cloud-based, ML-driven methodology that promotes the implementation of AI-enhanced precision farming via complimentary, open-access platforms. However, the authors used only ML models and the cloud-only system and did not consider the latency and energy consumption [20]. Bakthavatchalam et al. [13] proposed an IoT-enabled, ML-driven crop recommendation system that utilized real-time environmental data, benefiting from immediate data capture and open-source technologies such as WEKA. However, the authors used only ML models and did not consider the energy efficiency issue [13]. Cedric et al. [14] used three ML models to forecast the annual yields of six principal crops across nine

West African nations, employing cross-validation to mitigate overfitting. The restricted crop choices and dependence on fundamental ML models limit its wider applicability and capacity to manage more intricate agricultural data [14]. Further, the latency and energy consumption were not considered in [14].

We observe that most of the existing approaches ([13], [14], [20]) considered only ML models. Further, none of the existing approaches ([7], [13], [14], [20], [22], [23]) consider the latency and energy issues which are crucial for real-time applications. Further, there are some limitations, such as insufficient privacy mechanisms, limited scalability, limited crop selection, limited dataset, dependence on centralized data processing that heightens latency and energy expenses, inadequate capacity to capture temporal dependencies in agricultural data, etc. In our proposed approach, we use FL for privacy preservation, deep learning (DL) for large-scale data analysis, and edge computing for low latency and energy efficiency. FLyer integrates FL with edge computing to facilitate local data processing while ensuring privacy. Furthermore, the application of LSTM models in both local and global contexts effectively captures the sequential characteristics of crop data, thereby enhancing prediction accuracy and efficiency. Table I presents a comparison between FLyer and existing crop yield prediction systems, along with the pros and cons of each approach. Dew computing enables temporary data storage inside the user devices and edge computing with FL mitigates issues such as latency and energy consumption. The gradient encryption during model parameters' exchange, and data encryption during local storage protects data privacy. The exchange of model parameters between the edge nodes and the cloud, collaborative learning, and subsequent adjustment of the global model, improve the accuracy of predictions.

III. FLyer Framework

FLyer integrates edge computing, FL, DL-based data analytics, and data encryption, for crop yield prediction in Agriculture 5.0. In FLyer, LSTM is used as the DL model for data analysis, which is improved time-to-time with more data arrival. In FLyer, the edge nodes serve as clients, which perform local data processing, and the cloud acts as the server, where the global model is executed.

A. Proposed Architecture

The principal elements of FLyer are sensors with microcontrollers, user devices, edge servers, and private cloud servers. Fig. 1 pictorially demonstrates the four-tier architecture of FLyer, described as follows.

- Tier-I is the *IoT layer* that contains sensors and microcontrollers as the IoT devices. A sensor (ζ) is represented by a three tuple:
$$\zeta =< \zeta_{id}, \zeta_m, \zeta_o >$$
where ζ_{id} denotes ID of the sensor node, ζ_m denotes its mode (active or idle), and the type of the object is represented by ζ_o. The sensors collect soil data such as nitrogen (N), phosphorus (P), potassium (K), pH, and environmental data such as rainfall, temperature, and transmit the data to the connected microcontroller. A microcontroller (η) is represented by a three-tuple:
$$\eta =< \eta_{id}, \eta_m, \eta_{sp} >$$
where η_{id}, η_m, and η_{sp} denotes the microcontroller ID, its mode, and its configuration in terms of memory, processor, etc., respectively. The microcontroller receives data from the sensors and sends it to the user's device present in tier-II.
- Tier-II contains the user devices, such as mobile phones, tablets, etc. A user device (μ) is represented as a three tuple:
$$\mu =< \mu_{id}, \mu_m, \mu_{cf} >$$
where μ_{id} denotes ID of the user device, μ_m denotes its mode, and the configuration (processor, memory, etc.) is represented by μ_{cf}. The user device pre-processes the data after receiving from the microcontroller, and holds the data when Internet connectivity is not present. The pre-processed data is transmitted to the corresponding edge server in tier-III, when the Internet connectivity is available.
- Tier-III represents the *edge layer* that has the edge servers. An edge server (ξ) is represented by a three-tuple:
$$\xi =< \xi_{id}, \xi_m, \xi_{cf} >$$
where ξ_{id} represents the ID of the edge server, the mode is represented by ξ_m, and ξ_{cf} represents the configuration (processor, storage, memory, etc.). The edge server receives pre-processed data from the user device, and then locally processes the data. In FLyer, local models are trained on each edge server using distinct local datasets. The edge servers transmit their model updates to the cloud in tier-IV. For gradient encryption, we consider Advanced Encryption Standard (AES), and for data encryption during local storage we consider RSA and AES. AES is a popular private key cryptography algorithm used for gradient and data encryption purposes in FLyer. RSA algorithm is a popular public key cryptography algorithm used for data encryption purpose.
- Tier-IV represents the *cloud layer* that contains the cloud servers. The connection between the edge server and the cloud is crucial in the system architecture. Once training is completed, the updated weighted parameters from each edge server are sent to the cloud for aggregation, and the global model in the cloud is then updated. The updated model parameters are then transmitted back to the edge servers from the cloud. Notably, gradient encryption is applied during the exchange of model parameters, to ensure data security. A cloud server instance is represented by a two-tuple:
$$\tau =< \tau_{id}, \tau_{pr} >$$
where τ_{id} represents the ID of the cloud server instance and τ_{pr} represents the set of the IDs of the processing units.

TABLE I
COMPARISON AMONG FLYER AND EXISTING CROP YIELD PREDICTION SYSTEMS

Work	Model used	Pros	Cons	Used Edge computing	FL-centric system	Measured Latency	Measured Energy consumption	Used Encryption
Idoje et al. [7]	Hyper-tuned federated averaging	Privacy preservation	Limited dataset	✓	✓	✗	✗	✗
Bakthavatchalam et al. [13]	ML models	Real-time data acquisition	Limited scalability analysis, privacy is not considered	✗	✗	✗	✗	✗
Cedric et al. [14]	ML models	Cross validation to avoid overfitting	Limited crop selection	✗	✗	✗	✗	✗
Cruz et al. et al. [15]	ML model	Real-time data	Limited scalability analysis, privacy is not considered	✗	✗	✗	✗	✗
Kathiria et al. et al. [16]	ML models	Real-time data	Limited scalability analysis, privacy is not considered	✗	✗	✗	✗	✗
Gopi et al. et al. [17]	Deep learning models	Real-time data	Privacy is not considered	✗	✗	✗	✗	✗
Thilakarathne et al. [20]	ML models	Real-time data	Limited scalability analysis, privacy is not considered	✗	✗	✗	✗	✗
Zhang et al. [22]	Federated breeding	Privacy preservation, Emperical validation	Limited clients	✗	✓	✗	✗	✓
Reyana et al. [23]	Multisensor Machine-Learning	Multisensor data fusion	Crop-specific	✗	✗	✗	✗	✗
FLyer (Proposed)	FL using DL	Privacy preservation, low latency, low energy	Only CFL is considered	✓	✓	✓	✓	✓

B. Data Analysis using FL

The edge server-side process for local model update is stated in Algorithm 1, and the cloud server-side process for global model update is stated in Algorithm 2. Each edge server has its local dataset, which is trained locally. Each edge server transmits the local model updates to the cloud that performs aggregation using FAV function of Algorithm 2. The cloud server modifies the global model after receiving all the updates and sends the revised model updates to the connected edge servers after encrypting the gradients. In the algorithms, ξ denotes an edge server, N_ξ represents the number of edge servers connected with the cloud, P_ξ represents the proportion of edge servers that are involved in the FL process, N_r denotes the number of rounds, θ denotes the cloud, β denotes the batches, N_e denotes the number of epochs, the local dataset of ξ is denoted by $Data_\xi$, N_β denotes the number of batches, and during training the dataset is split into N_β batches.

In FLyer, LSTM is used in global model as well as in local model. LSTM is a gated Recurrent Neural Network (RNN) with memory cells to deal with sequential data, capable of learning long term dependencies, and forget unrelated information. In predicting crop yield, different types of environment and soil-related data are often combined as input features for the prediction model. Especially throughout the crop growing season, these input features change over time. The LSTM model can capture the temporal dependencies and hold the sequential nature of the data.

The training of an LSTM model follows several steps with the sequence of inputs x_t, computed hidden state (h_t), and output sequences. In forward propagation, formulation of hidden layers of the LSTM for each time step t with forget gate (Gf_t), input gate (Gi_t), output gate (Go_t), candidate gate (Gc_t), new cell state (Sc_t), and hidden state (h_t), are

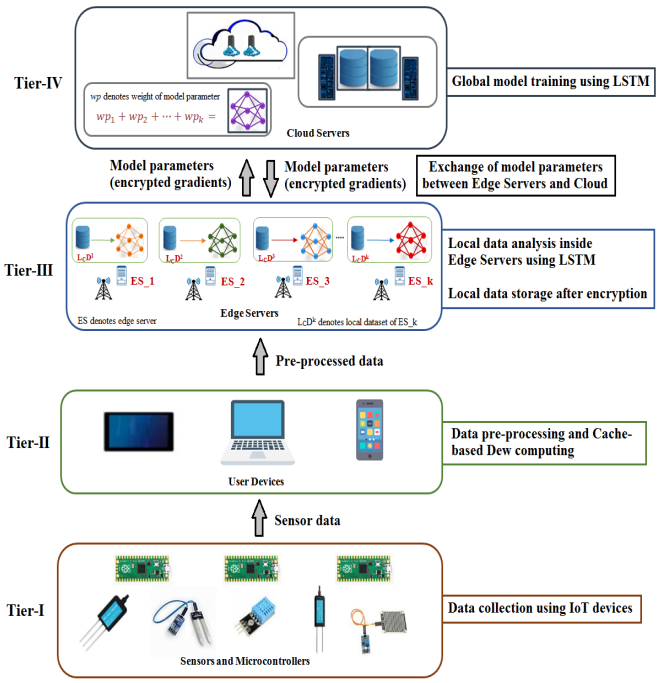

Fig. 1. Four-Tier architecture of FLyer

Algorithm 1 Local model update inside the edge server

Input: $Data_\xi$, β, N_β, N_e
Output: Model parameters (M_{up})

$Function\ Update(Data_\xi, \beta, N_\beta, N_e)$:
$Data_{pr} \leftarrow Process(Data_\xi)$
while ($connection\ between\ \xi\ and\ \theta\ is\ present$) **do**
  collect M_{colen} ▷ M_{colen} denotes model parameters with encrypted gradients
  $M_{col} \leftarrow decryptgradient(M_{colen})$
  $M_{upn} \leftarrow FLAvg(M_{up})$
end while
$SaveParam(M_{upn})$
$M_{en} \leftarrow encryptgradient(M_{upn})$
return M_{en} to θ
$Function\ FLAvg(M_{up}, \beta, N_\beta, N_e)$:
$\beta \leftarrow Split(Data_{pr}, N_\beta)$
for $e = 1$ *to* N_e **do**
  for $b \in \beta$ **do**
    $M_{up} \leftarrow M_{up} - LR * Gradient(M_{up}, b)$ ▷ LR denotes learning rate
  end for
end for

mathematically presented as follows:

$$Gf_t = \sigma(w_{Gf} h_{t-1} + w_{Gf} x_t + bias_{Gf}) \quad (1)$$

$$Gi_t = \sigma(w_{Gi} h_{t-1} + w_{Gi} x_t + bias_{Gi}) \quad (2)$$

$$Gc_t = \tanh(w_{Gc} h_{t-1} + w_{Gc} x_t + bias_{Gc}) \quad (3)$$

$$Sc_t = (Gi_t * Gc_t) + (Gf_t * Sc_{t-1}) \quad (4)$$

$$Go_t = \sigma(w_{Go} h_{t-1} + w_{Go} x_t + bias_{Go}) \quad (5)$$

$$h_t = Go_t * \tanh(Sc_t) \quad (6)$$

where h_{t-1} represents the hidden state at time $(t-1)$. $bias_{Gi}$ is the bias for the input gate, $bias_{Gf}$ is the bias for the forget gate, $bias_{Gc}$ is the bias for the candidate gate, and $bias_{Go}$ is the bias for the output gate, w_{Gi} is the weight matrix for the input gate, w_{Gf} is the weight matrix for the forget gate, w_{Gc} is the weight matrix for the candidate gate, and w_{Go} is the weight matrix for the output gate. The weighted parameters are revised during the training. For classification using SoftMax function, the output layer y_t is expressed as:

$$y_t = SoftMax(wh_t + bias_t) \quad (7)$$

Table II presents the LSTM model features with their values used for classification in FLyer. As the first and second dense layer activation function ReLU is used. Sparse Categorical Crossentropy (SCC) is used as the loss function.

1) Time complexity: To train using local dataset, the time complexity is given by, $O(N_{e_{loc}} \cdot n_\xi \cdot (u_1 \cdot u_2 + u_2 \cdot u_3 + \ldots + u_{U-1} \cdot u_U))$, where the number of local epochs is denoted by $N_{e_{loc}}$, the number of local data samples for ξ is denoted as n_ξ where $\xi \in \xi_{tot}$, ξ_{tot} is the set of edge servers, N_ξ is the number of edge servers, u_i denotes i^{th} layer, and the number of layers in the neural network is represented by U. The global model's time consumption is dependent on N_ξ i.e. the number of edge servers and M that represents the aggregated model parameters. For exchanging parameters, the time consumption is dependent on M. Therefore, if both the local and global models are incorporated, the time complexity

Algorithm 2 Global model update inside the cloud

Input: N_ξ, P_ξ, N_r
Output: Model parameters (M_r) at the end of all rounds

$Function\ Gather(N_\xi, P_\xi, N_r)$:
while $k = 1$ *to* N_ξ **do**
  receive model parameters from k
  $M_{up} \leftarrow decryptgradient(M_{en})$
end while
$FAV(N_\xi, P_\xi, N_r)$
release edge servers which are connected
$Function\ FAV(N_\xi, P_\xi, N_r)$:
$M_1 \leftarrow InitModel()$ ▷ model parameters are initialized
for $r = 1$ *to* N_r **do**
  $\xi_r \leftarrow Subset(max(P_\xi * N_\xi, 1), \text{“}random\text{”})$
  for $k = 1$ *to* N_{ξ_r} **do** ▷ N_{ξ_r} denotes number of elements in ξ_r
    $M^k_{r+1} \leftarrow FLAvg(M^k)$ ▷ call $FLAvg$ function from Algorithm 1
  end for
  $M_{r+1} \leftarrow \sum_{k=1}^{N_{\xi_r}} \frac{size(Data_k)}{size(Data_{N_{\xi_r}})} \cdot M^k_{r+1}$ ▷ $Data_k$ denotes local dataset of k, $Data_{N_{\xi_r}}$ denotes total dataset of all edge servers
end for

TABLE II
THE VALUES OF THE PARAMETERS FOR MODEL CLASSIFICATION

Classifier	Feature	Value
LSTM	LSTM unit:	64
	First Dense layer unit:	128
	Dropout:	0.2
	Second Dense layer unit:	64
	Dropout:	0.4
	Output layer unit:	2
	Optimizer:	Adam

is given by equation (8).

$$T(FLyer) = O(N_\xi \cdot (N_{e_{loc}} \cdot n_\xi \cdot (u_1 \cdot u_2 + u_2 \cdot u_3 + \ldots + u_{U-1} \cdot u_U))) + O(N_\xi.M) + O(M) + T(Crypt_{grad}) + T(Crypt_{data}) \quad (8)$$

where $T(Crypt_{grad})$ denotes the time complexity for gradient encryption and decryption, and $T(Crypt_{data})$ denotes the time complexity for data encryption and decryption. The selection of the cryptography algorithm will depend on the user. In FLyer, we consider AES for gradient encryption, and for data encryption during local storage we consider AES and RSA.

The time complexity for gradient encryption is $O(AES_{Gradbits})$, where $AES_{Gradbits}$ denotes the size of the gradient information in bits. Then, $T(Crypt_{grad}) = O(AES_{Gradbits})$. If RSA is used for data encryption, $T(Crypt_{data}) = O(RSA_{key_{data}})$, where $RSA_{key_{data}}$ denotes the key size used for data encryption using RSA. If AES is used for data encryption, $T(Crypt_{data}) = O(AES_{data})$, where AES_{data} denotes the data size in bits.

2) Training Energy Consumption: The training energy is obtained by adding the energy consumed by the cloud (E_{cloud}) with the energy consumed by the edge servers (E_{edge}), as follows [6]:

$$E_{FLyer} = E_{cloud} + E_{edge} \quad (9)$$

E_{cloud} and E_{edge} are determined by equations (10) and (11) as follows.

$$E_{cloud} = \sum_{r=1}^{N_r} (t_{\theta_r} \cdot e_{\theta_r}) \quad (10)$$

and

$$E_{edge} = \sum_{r=1}^{N_r} \sum_{k=1}^{N_\xi} (t_{k_r} \cdot e_{k_r}) \quad (11)$$

where N_r represents the number of rounds, N_ξ represents the number of edge servers, t_{θ_r} represents the time consumption for cloud (θ) at round r, e_{θ_r} represents the energy consumed by the cloud (θ) at round r, t_{k_r} represents the time consumed by edge server k at round r, and e_{k_r} represents the energy consumed by edge server k at round r.

C. Latency in FLyer

To determine the latency, the communication latency and computational latency for data analysis are considered. The communication latency for transmitting data from node j to node q is determined as $(1 + \phi_{jq}) \cdot \frac{data_{jq}}{\rho_{jq}}$, where ϕ_{jq} denotes the rate of link failure, $data_{jq}$ denotes the data amount, and ρ_{jq} denotes the rate of data transmission [2], [6]. The computational time for analyzing data by a node j is determined as $\frac{data_j}{speed_j}$, where $data_j$ represents the data amount and $speed_j$ represents the processing speed of node j [2], [6].

Let, the data pre-processing latency is L_{pre}, the latency for transmitting data from tier-II to tier-III is denoted as $L_{\mu\xi}$, the latency for local data analysis in tier-III is $L_{p\xi}$, the latency for model parameters' exchange between tier-III and tier-IV is $L_{\xi\tau}$, the latency for global model update in tier-IV is $L_{p\tau}$, the time consumption for gradient encryption and decryption is L_{gen}, and the time consumption for data encryption and decryption is L_{den}. Then, the total latency of FLyer is determined by equation (12) as follows:

$$L_{FLyer} = L_{pre} + L_{\mu\xi} + L_{p\xi} + L_{\xi\tau} + L_{p\tau} + L_{gen} + L_{den} \quad (12)$$

The performance of FLyer is evaluated in Section IV from the perspective of prediction accuracy, latency, and consumption of energy.

IV. Performance Evaluation

In the realm of data analysis, our investigation relies on a dataset[1] that contains crop-oriented information across districts of Western Maharashtra, India. The collection comprises a detailed study of 4513 samples. The dataset includes essential factors such as Soil colour, Nitrogen (N), Potassium (K), Phosphorus (P), pH, Rainfall, and Temperature. Based on the parameters the suitable crop for the land is predicted. Table III presents a statistical summary of the quantitative parameters of the dataset.

TABLE III
Statistical Summary of the Dataset Used for Analysis

Parameter	Minimum value	Maximum value	Mean	Standard deviation
N	20	150	95.41	38.06
P	10	90	54.34	16.55
K	5	150	63.595	35.69
pH	5.5	8.5	6.715	0.625
Rainfall	300	1700	819.189	251.73
Temperature	10	40	25.915	5.897

A. Test-bed for Experiment

For data analysis, we have used FL. The data is locally analyzed inside the edge servers and global model training occurs inside the cloud servers. We have considered four edge servers (RAM: 16 GB, HDD: 1TB, processor: Intel(R) Xeon(R) CPU E5-2667 0 @ 2.90GHz (Octa Core)). For cloud services, private cloud servers are used (RAM: 16 GB, HDD: 2TB, processor: Intel(R) Xeon(R) CPU ES-2667 0 @ 2.90 GHz (Hexa Core)). The python is used for implementation, and for the deep learning TensorFlow Enterprise 2.3 (CUDA 11.0) is used. In our testbed, we have considered the edge servers and private cloud servers within the same Local area network. For exchanging model parameters between an edge server and the private cloud server, we have used socket programming and TCP client-server model. The experimental setup is presented in Fig. 2. The dataset is split into five parts. The four parts are assigned as the local datasets to the edge servers, and one part is assigned to the cloud as the global dataset.

1) Prediction accuracy: The assessment metrics for the local and global models in FLyer are presented in Table IV and Fig. 3. The confusion matrices for the local models are presented in Figs. 4 to 7. The confusion matrix for the global model is presented in Fig. 8. The prediction accuracy achieved

[1] https://www.kaggle.com/datasets/sanchitagholap/crop-and-fertilizer-dataset-for-westernmaharashtra

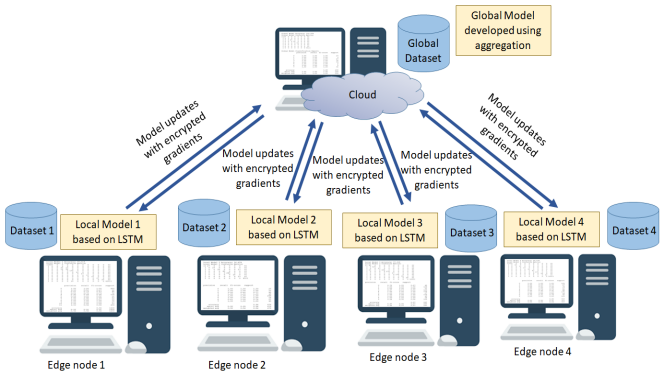

Fig. 2. Experimental setup with four edge nodes and cloud

TABLE IV
PERFORMANCE OF LSTM IN FLYER

Model	Accuracy	Precision	Recall	F1-score
Local model 1	0.9601	0.93	0.91	0.91
Local model 2	0.9645	0.95	0.95	0.94
Local model 3	0.9732	0.97	0.98	0.97
Local model 4	0.9463	0.92	0.9	0.91
Global model	>0.99	>0.99	>0.99	>0.99

by the local models for the edge servers 1, 2, 3, and 4 are 96.01%, 96.45%, 97.32%, and 94.63%, respectively. The precision, recall, and F1-score for the edge servers 1, 2, 3, and 4 are $\geq$0.9, i.e. 90%. The average prediction accuracy, precision, recall, and F1-score for the local models are 96.1%, 94.25%, 93.5%, and 93.25%, respectively. The results demonstrate that the crop yield prediction accuracy achieved by the global model in the proposed framework FLyer is above 99%. The high level of crop yield prediction accuracy achieved by the FLyer highlights its effectiveness in producing precise and dependable outcomes within the specified framework.

2) Latency: From the experimental results, we have observed that the latency for gradient encryption and decryption

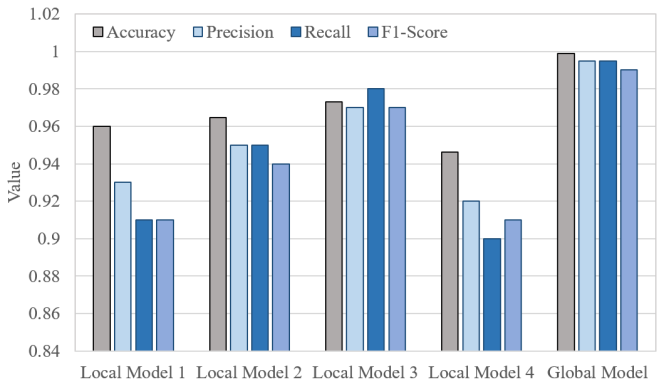

Fig. 3. Performance of the local and global models in FLyer

using AES-256 is 4.29 ms. From the results, we have also observed that the latency for data encryption and decryption during local storage using RSA and AES-256 are 145s and 39.08 ms respectively. We observe that AES-256 consumes less time for data encryption than RSA. The time consumption for data encryption and decryption is not very high in AES, but privacy can be protected. Hence, during local storage AES can be used to encrypt the data to protect data privacy. As FL is used, the raw data is not shared; further, gradients are encrypted using AES in FLyer. The results show that the time consumed for gradient encryption and decryption using AES is low. Hence, during model parameters' exchange gradient encryption is recommended to reduce the gradient information leakage ratio. This is observed that the average gradient information leakage ratio for 20%, 40%, and 50% malicious participants are 0.002, 0.02, and 0.038, respectively, which is very low. The total latency considering data transmission and analysis for FLyer, edge-cloud-based framework without FL, and cloud-only framework are determined and presented in Fig. 9. The average data transmission speed of uplink has been 5 Mbps and downlink has been 10 Mbps. For the considered scenario, FLyer, edge-cloud framework without FL, and cloud-only framework have the latency of 10-15s, 20-25s, and 30-35s, respectively. The results demonstrate that FLyer achieves a reduction of 39% and 56% latency than the edge-cloud-based framework without FL, and cloud-only framework, respectively.

3) Energy consumption: The training energy consumption for FLyer, edge-cloud-based framework without FL, and cloud-only framework are determined and presented in Fig. 10. For the considered scenario, FLyer, edge-cloud framework without FL, and cloud-only framework have the energy consumption of 4-6KJ, 8-10KJ, and 12-14KJ, respectively. This is observed that FLyer reduces approximately 40% and 57% energy consumption than edge-cloud-based framework without FL, and cloud-only framework, respectively.

4) Comparison with existing frameworks: FLyer is compared with the existing approaches on crop yield prediction in Table V. In [7], FL was used and Gaussian Naive Bayes was used as the classifier, and the achieved accuracy was 90%. In [13], MLP was used as the classifier, and the achieved accuracy was 98.23%. In [20], RF, DT, KNN, XGBoost, and SVM were used as the classifiers, and among them RF achieved the highest accuracy of 97.18%. In [23], DT, Hoeffding tree, and RF, were used, and the achieved accuracy was 91.25%. In [13], [20], and [23], FL was not used. The results demonstrate that the accuracy in FLyer is higher than the existing schemes [7], [13], [20], and [23]. This is also seen that the existing approaches did not measure the latency and energy consumption, though energy consumption and latency both are vital for sustainable real-time systems. As FL is used in FLyer, no raw data sharing takes place. Thus, data privacy is protected in FLyer. Moreover, the use of encryption in local data storage and gradient encryption during model parameters' exchange, enhance data security. This is also observed that the gradient information leakage ratio is only 0.038 for 50% malicious participants in FLyer. Therefore, we can conclude that FLyer is better than the state-of-the-art.

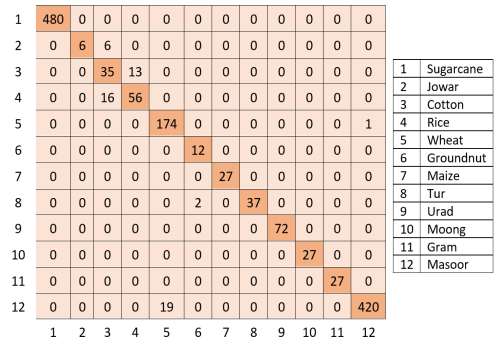

Fig. 4. Confusion matrix for local model 1

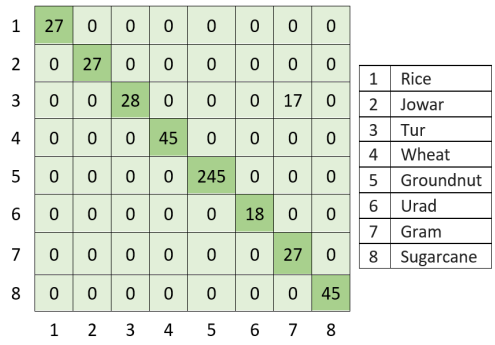

Fig. 5. Confusion matrix for local model 2

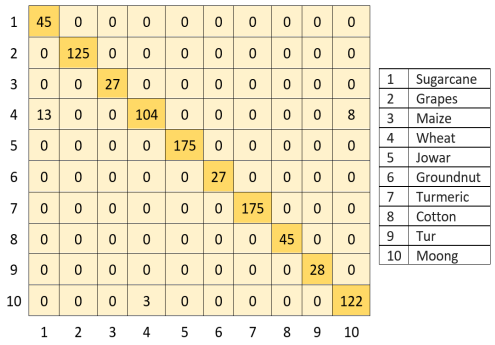

Fig. 6. Confusion matrix for local model 3

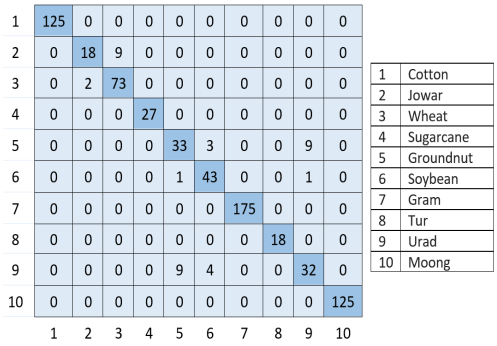

Fig. 7. Confusion matrix for local model 4

TABLE V
PERFORMANCE COMPARISON AMONG FLYER AND EXISTING CROP YIELD PREDICTION FRAMEWORKS

Work	Classifier	FL is used	Accuracy	Latency	Energy consumption	Gradient information leakage ratio
Idoje et al. [7]	Gaussian Naive Bayes	Yes	90%	Not determined	Not determined	Not determined
Bakthavatchalam et al. [13]	Multilayer Perceptron (MLP)	No	98.23%	Not determined	Not determined	Not determined
Thilakarathne et al. [20]	RF (Highest), DT, KNN, XGBoost, SVM	No	97.18%	Not determined	Not determined	Not determined
Reyana et al. [23]	DT, Hoeffding tree, RF	No	91.25%	Not determined	Not determined	Not determined
FLyer	LSTM	Yes	>99%	2.43	Yes	0.038 (for 50% malicious participants)

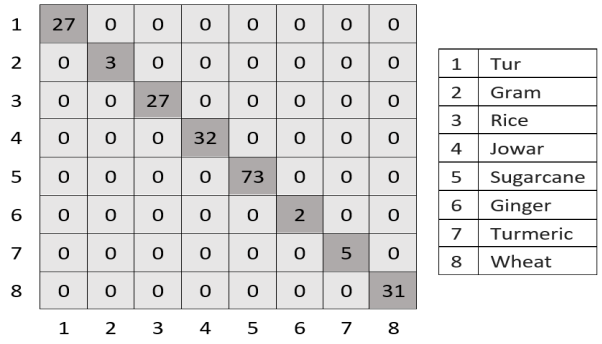

Fig. 8. Confusion matrix for the global model

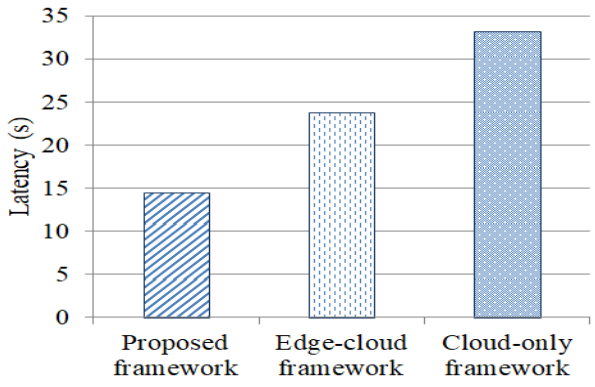

Fig. 9. Comparison of latency in FLyer (proposed), edge-cloud framework without FL, and cloud-only framework

V. CONCLUSIONS AND FUTURE WORK

This paper has proposed a crop yield prediction framework using FL, referred to as FLyer. The soil and environmental data

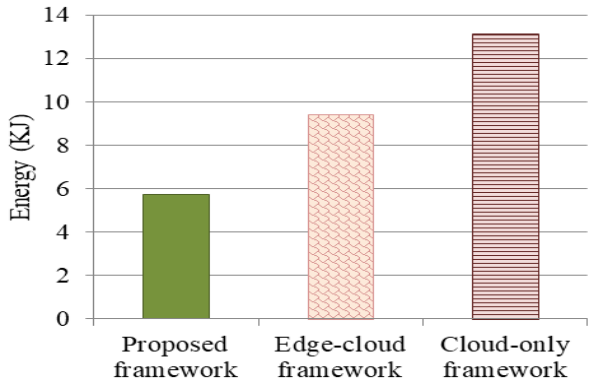

Fig. 10. Comparison of energy consumption in FLyer (proposed), edge-cloud framework without FL, and cloud-only framework

are processed locally inside the edge servers using LSTM. The model parameters are exchanged between the edge servers and the cloud. No raw data sharing is performed, further, model parameters are exchanged after encrypting gradients. The use of encryption during model parameters' exchange reduces gradient information leakage ratio to 0.038 for 50% malicious participants. Further, during local data storage encryption is performed. We have used AES-256 for gradient encryption. For data encryption during local storage, we have used RSA and AES-256, and observed that AES-256 consumes lesser time. The use of gradient encryption-based FL improves data privacy, and the data is locally processed inside the edge servers to reduce latency and energy consumption. From the results we have observed that the global model in FLyer has achieved above 99% accuracy in crop yield prediction. We have also observed that FLyer reduces the latency by $\sim$39% and energy consumption by $\sim$40% than the edge-cloud framework without FL respectively.

The FLyer architecture improves model explainability by utilizing LSTM networks to capture temporal relationships that is crucial for crop yield prediction, while FL maintains data privacy by processing information locally and exchanging only model parameters with encrypted gradients. The use of AES-256 for gradient and data encryption achieves a balance between security and computing efficiency, making the model both secure and efficient. FLyer's operations are sustainable and comprehensible for agricultural applications due to lowered latency and energy consumption. Future initiatives may enhance explainability by incorporating generative AI to produce synthetic data for more comprehensive local model training, alongside employing Bayesian LSTM to provide probabilistic insights, thereby reducing the limitations for end-users.

REFERENCES

[1] V. Saiz-Rubio and F. Rovira-Más, "From smart farming towards agriculture 5.0: A review on crop data management," *Agronomy*, vol. 10, no. 2, p. 207, 2020.

[2] 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," *The Journal of Supercomputing*, pp. 1–35, 2023.

[3] W.-J. Chang, L.-B. Chen, C.-Y. Sie, and C.-H. Yang, "An artificial intelligence edge computing-based assistive system for visually impaired pedestrian safety at zebra crossings," *IEEE Transactions on Consumer Electronics*, vol. 67, no. 1, pp. 3–11, 2020.

[4] A. E. Alchalabi, S. Shirmohammadi, S. Mohammed, S. Stoian, and K. Vijayasuganthan, "Fair server selection in edge computing with q-value-normalized action-suppressed quadruple q-learning," *IEEE Transactions on Artificial Intelligence*, vol. 2, no. 6, pp. 519–527, 2021.

[5] A. Mukherjee, D. De, N. Dey, R. G. Crespo, and E. Herrera-Viedma, "Disastdrone: A disaster aware consumer internet of drone things system in ultra-low latent 6g network," *IEEE Transactions on Consumer Electronics*, vol. 69, no. 1, pp. 38–48, 2022.

[6] S. Bera, T. Dey, A. Mukherjee, and D. De, "Flag: Federated learning for sustainable irrigation in agriculture 5.0," *IEEE Transactions on Consumer Electronics*, 2024.

[7] G. Idoje, T. Dagiuklas, and M. Iqbal, "Federated learning: Crop classification in a smart farm decentralised network," *Smart Agricultural Technology*, vol. 5, p. 100277, 2023.

[8] 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*, vol. 165, pp. 17–31, 2022.

[9] Y. Wang, Y. Zhou, and P.-Q. Huang, "A novel incentive mechanism for federated learning over wireless communications," *IEEE Transactions on Artificial Intelligence*, vol. 5, no. 11, pp. 5561–5574, 2024.

[10] S. M. Rajagopal, M. Supriya, and R. Buyya, "Fedsdm: Federated learning based smart decision making module for ecg data in iot integrated edge-fog-cloud computing environments," *Internet of Things*, p. 100784, 2023.

[11] K. Huang, X. Liu, F. Li, C. Yang, O. Kaynak, and T. Huang, "A federated dictionary learning method for process monitoring with industrial applications," *IEEE Transactions on Artificial Intelligence*, 2022.

[12] D. De, S. Ghosh, and A. Mukherjee, "Socialsense: Mobile crowd sensing-based physical distance monitoring model leveraging federated learning for pandemic," *Internet of Things*, vol. 23, p. 100872, 2023.

[13] K. Bakthavatchalam, B. Karthik, V. Thiruvengadam, S. Muthal, D. Jose, K. Kotecha, and V. Varadarajan, "Iot framework for measurement and precision agriculture: predicting the crop using machine learning algorithms," *Technologies*, vol. 10, no. 1, p. 13, 2022.

[14] L. S. Cedric, W. Y. H. Adoni, R. Aworka, J. T. Zoueu, F. K. Mutombo, M. Krichen, and C. L. M. Kimpolo, "Crops yield prediction based on machine learning models: case of west african countries," *Smart Agricultural Technology*, p. 100049, 2022.

[15] M. Cruz, S. Mafra, and E. Teixeira, "An iot crop recommendation system with k-nn and lora for precision farming," 2022.

[16] 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, Volume 1*. Springer, 2023, pp. 841–850.

[17] P. Gopi and M. Karthikeyan, "Red fox optimization with ensemble recurrent neural network for crop recommendation and yield prediction model," *Multimedia Tools and Applications*, vol. 83, no. 5, pp. 13 159–13 179, 2024.

[18] 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.

[19] M. Cordeiro, C. Markert, S. S. Araújo, N. G. Campos, R. S. Gondim, T. L. C. da Silva, and A. R. da Rocha, "Towards smart farming: Fog-enabled intelligent irrigation system using deep neural networks," *Future Generation Computer Systems*, vol. 129, pp. 115–124, 2022.

[20] 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*, vol. 22, no. 16, p. 6299, 2022.

[21] T. Manoj, K. Makkithaya, and V. Narendra, "A federated learning-based crop yield prediction for agricultural production risk management," in *2022 IEEE Delhi Section Conference (DELCON)*. IEEE, 2022, pp. 1–7.

[22] Q. Zhang, X. Zhao, Y. Han, F. Yang, S. Pan, Z. Liu, K. Wang, and C. Zhao, "Maize yield prediction using federated random forest," *Computers and Electronics in Agriculture*, vol. 210, p. 107930, 2023.

[23] A. Reyana, S. Kautish, P. S. Karthik, I. A. Al-Baltah, M. B. Jasser, and A. W. Mohamed, "Accelerating crop yield: Multisensor data fusion and machine learning for agriculture text classification," *IEEE Access*, vol. 11, pp. 20 795–20 805, 2023.

[24] H. Yang, H. Yin, F. Li, Y. Hu, and K. Yu, "Machine learning models fed with optimized spectral indices to advance crop nitrogen monitoring," *Field Crops Research*, vol. 293, p. 108844, 2023.