Rice Leaf Disease Prediction Using Light Weight Federated Learning

Asis Asutosh Sahoo[1], T Prachi Patro[2], Ashish Kumar Dass[3]

Department of Computer Science and Engineering, NIST University, Berhampur, Odisha-761008, India

Abstract

Worldwide, rice stands as an essential staple crop with multiple leaf diseases which affect its yield yet remain undetected during initial stages of development. The traditional methods for diagnosing diseases depend on experts who need to conduct manual inspections, which results in slow and inaccurate processes that cannot handle operations of extensive agricultural fields. Deep-learning methods have proven to be highly effective for detecting plant diseases, but these methods need all data to be gathered in one place. The situation creates major problems because it involves protecting data privacy, needing continuous internet access, causing more data transmission since users must provide extra information, and exposing systems to potential cyberattacks. The project develops a Lightweight Federated Learning (LWFL) framework to predict rice leaf disease which allows training on various edge devices without sending actual data to main servers. The proposed system uses a pre-trained InceptionV3 Convolutional Neural Network (CNN) model which undergoes optimization through pruning and data augmentation and image normalization techniques to achieve reduced computational requirements while maintaining precise results. The Flower federated learning framework handles client communication, uses Federated Averaging (FedAvg) to combine model updates, and supports decentralized learning. The dataset is distributed among simulated clients to mimic realworld conditions where each farmer or edge node holds its own local data. The optimized federated model reaches approximately 98.5% accuracy, while showing strong results in both macro and weighted precision and recall and F1-score performance according to the experimental results. The system maintains its capability to perform under conditions of data diversity and resource limitations on client systems. The project demonstration shows that lightweight federated learning models can be used for agricultural disease detection because they provide privacy protection and efficient resource use while offering scalable AI solutions which work in both rural areas and actual farming operations.

Keywords : Deep-learning, LWFL, CNN, Federated Averaging, Pruning, Data Augmentation, InceptionV3, F1 Score

Introduction

Developing countries rely on agriculture as their fundamental economic support system because rice serves as the basic food that provides sustenance to more than half of the world's population. The rising demand together with climate change effects creates difficulties for scientists who must sustain rice production while reducing operational losses. Leaf diseases represent a major threat to rice farming because they have the potential to destroy both plant development and agricultural yield. The traditional approach for identifying diseases depends on inspecting plants with human eyes, which requires extensive time and physical work but results in frequent mistakes. The field of plant disease detection has progressed because of computer vision and machine learning (ML) advances, which combine to create automated systems that use image classification for disease identification. The systems require organizations to gather extensive labeled image datasets, which they use to teach deep learning algorithms that include Convolutional Neural Networks (CNNs) before using the trained models for inference in their mobile or web systems. Centralized training creates distinct problems that include safeguarding data privacy together with the requirement of needing extensive computing power. The development of Federated Learning (FL) provides a solution that helps to overcome these challenges. The Federated Learning (FL) system enables multiple clients like edge devices and users to work together in developing a common global model which does not require them to send their original data to a central location. The system only transmits model updates which enable user data to remain confidential. The FL system enables training without revealing

user information, but the implementation of advanced models like ResNet and Inception requires too much computing power for edge devices. Lightweight Federated Learning (LWFL) solves the problem through model compression techniques which include pruning and quantization to decrease model size and operational demands while preserving system efficiency. The system enables FL operations on devices with limited power capabilities which makes it suitable for agricultural applications that require rice disease prediction in remote and field environments. The internship project aims to develop an LWFL solution which will detect various rice leaf diseases through analysis of a specially selected dataset. The system achieves high prediction accuracy through its compact model design which ensures strong data protection making it suitable for use in actual agricultural settings.

Literature review

The use of Artificial Intelligence in agriculture currently receives extensive research coverage because scientists developed advanced techniques to identify plant diseases. Farmers and agricultural experts must conduct manual inspections to detect crop diseases through traditional methods. The methods show some effectiveness which requires extensive time for processing and leads to subjective outcomes that fail to deliver accurate results in large-scale farming operations.

Deep learning advancements have made Convolutional Neural Networks (CNNs) the main technology used for plant disease classification through image analysis. Multiple research studies have shown that crop disease detection reaches high accuracy levels when using VGGNet and ResNet and InceptionV3 architectural frameworks. The models enable automated disease diagnosis through their ability to extract complex visual features which include texture and color variations and lesion patterns found in leaf images.

Centralized machine learning systems operate through data collection from various sources which academic organizations use to store information on a central server for model training purposes. The system achieves high accuracy results but it creates multiple real-world implementation difficulties. Agricultural data which includes images that farmers collect contains information that should be treated as confidential or proprietary material. The transfer of such information to a central server creates problems which involve privacy rights and ownership claims and data protection issues. The limited internet availability in rural regions creates challenges that prevent complete data transmission across large distances.

The issues require resolution through the implementation of Federated Learning (FL) which provides a viable solution. The system enables multiple clients which include mobile devices and edge systems to work together for training a model without disclosing their raw data. The system transmits only model updates to the central server which results in better privacy protection and decreased communication needs. Recent research has shown that federated learning can achieve performance comparable to centralized models while preserving data locality. The standard federated learning models present computational challenges that agricultural applications face in their actual deployment. The development of Lightweight Federated Learning (LWFL) approaches focuses on achieving two goals which include reducing model size and decreasing computation needs. The industry employs model pruning and model compression and transfer learning as standard techniques to enhance model efficiency without decreasing accuracy. The project develops these advancements through InceptionV3 model integration with pruning techniques within a federated learning framework. The system brings together deep learning capabilities with privacy protection and operational effectiveness to enable its use in actual agricultural fieldwork.

Problem formulation and system requirements

This section should precisely define what Lightweight Federated Learning does, what information it consumes and produces, and which requirements it must satisfy to be useful and reliable in practice.

Formal problem definition

Rice serves as a vital food source in most countries around the world, but its yield suffers from multiple leaf diseases which include bacterial leaf blight and brown spot and leaf blast. The detection of these diseases needs to occur at an early stage because it protects crop yields and supports food production stability. Farmers in various areas face disease identification challenges because they do not have access to professional help and modern diagnostic equipment which results in incorrect and delayed disease diagnosis.

AI-based methods which currently exist for detecting diseases depend on a process that requires data to be collected and processed at a central location. The system faces multiple difficulties. First, farmers may be unwilling or unable to share their data due to privacy concerns. Second, transferring large volumes of image data requires stable internet connectivity which is often unavailable in rural areas. Third, centralized systems cannot provide effective coverage across various geographic locations because they depend on specific environmental factors and agricultural patterns of each region.

The main problem which this project needs to solve requires the development of a rice leaf disease detection system. The system needs to provide accurate results while maintaining efficient operation and safeguarding user privacy. The system needs to function in multiple locations without needing to gather data at a central point.

The project implements a federated learning system which allows multiple clients to develop a common model through local training with their individual datasets. The central server receives model updates from all clients which it combines to create a unified global model. The system uses model optimization techniques through pruning and lightweight architecture design to make the system operational in resource-limited environments.

The system development needs to achieve four specific requirements which include maintaining accurate classification results and protecting user data and minimizing transmission costs and providing efficient operation on edge computing devices.

Hardware System Requirements

The system offers flexible operation because it can function on both typical machines and machines with moderate processing power. The system requires specific hardware components to achieve its best performance according to these recommended specifications:

- Processor (Server): Intel i5
- RAM (Server): Minimum 8 GB (16 GB recommended)
- Storage (Both Server and Client): At least 2-3 GB free space (for dataset and models)
- Client Devices: Can include laptops, edge devices, or mobile systems

Software System Requirements

The project is implemented using Python and requires the following software and libraries:

- Operating System: Windows / Linux / macOS
- Python Version: 3.10 (recommended for compatibility)

The project needs these libraries and frameworks to operate correctly:

- TensorFlow (Deep Learning)
- tf_keras (Keras compatibility layer)
- TensorFlow Model Optimization Toolkit (TFMOT) for pruning
- Flower (FLWR) for federated learning
- NumPy, Pandas (data handling)
- Matplotlib (visualization)
- Scikit-learn (evaluation metrics)

Functional Requirements

- The system needs to accomplish the following tasks:
- The system should load rice leaf image datasets from various client sources and proceed to preprocess them.
- The system should enable clients to conduct training on their individual devices.
- The system needs to implement data augmentation methods together with normalization methods.
- The system should use federated learning to enable multiple users to work together on training their shared model.
- The system should combine client updates through the implementation of the FedAvg algorithm.
- The system needs to use pruning methods for model optimization purposes.

- The central testing database will be used to assess the final worldwide model.
- The system will produce performance metrics which include accuracy, precision, recall, and F1-score.

Non-Functional Requirements

- Privacy: The system must restrict all raw data to remain on client devices.
- Scalability: The system must enable multiple clients to access its functions.
- Efficiency: The model needs to maintain a lightweight design which enables quick performance.
- The system must maintain full functionality during client partial outages.
- The system needs to provide simple installation and operation methods which require normal computer systems.

Dataset and Preprocessing

Overview of the Dataset

The dataset used for this project was obtained from an open-source repository hosted on Kaggle, titled ”Rice Leaf Disease Image Dataset”. This dataset contains high-resolution images which show rice leaves that have been infected by different common diseases and display healthy leaf samples. Each image is labeled according to one of the following nine classes:

1. Bacterial Leaf Blight
2. Brown Spot
3. Healthy Rice Leaf
4. Leaf Blast
5. Leaf Scald
6. Narrow Brown Leaf Spot
7. Neck Blast
8. Rice Hispa
9. Sheath Blight

The total dataset contains approximately 8,000 images which are distributed fairly evenly across these classes. Each image is in RGB format with an average resolution of 256×256 pixels.

Data Cleaning and Normalization

The federated learning system requires multiple preprocessing steps to complete data processing before starting the data feed. The system processed all images through image resizing which converted their original dimensions to 128 x 128 pixel sizes for two purposes. The system used one-hot encoding to convert class names into numerical labels which range from 0 to 8 for use with softmax output layers. The system converted pixel values to the [0 1] range through a division process that used 255 as the divisor. This process helps the model reach its goals faster while it decreases the model's output fluctuations. The system removed all duplicate images together with any existing corrupted files to prevent training processes from receiving biased data which would result in training errors.

Client Specific Dataset Distribution and Central Test Set

The researchers created a federated learning environment by dividing their complete dataset into five separate sections which each represented a different client. Each client received approximately 1,580–1,750 images. The researchers used stratified sampling to create client subsets which included images from all nine disease categories while maintaining class balance between clients. The researchers created client-specific datasets together with a central test set which contained 20% of the images from each class. The test set remained unused throughout the training process because it existed to assess the performance of the final aggregated model.

Data Augmentation

The system used data augmentation through its live operations which employed Random Horizontal and Vertical Flips to create different image orientations for testing. The system used Random Rotations (±20 degrees) to strengthen its defense against changes in observer positions. The experimentation uses Zoom and Shift

Transformations which introduce size and location changes to test the system's ability to handle small spatial variations. The training process created augmented data which prevented the model from encountering identical images so it would develop a broader range of training data.

Model Optimization

Base Model Selection: InceptionV3

The base model selection process plays a critical role in building an effective rice leaf disease classifier, which operates under federated learning conditions that create diverse computational capabilities among different users. For this project, InceptionV3 was chosen due to its excellent balance between accuracy, computational cost, and model size. InceptionV3, developed by Google, is a member of the Inception deep convolutional neural network family which has been specifically designed to achieve high performance in feature extraction tasks.

InceptionV3 uses Inception modules which let multiple filter sizes (1×1, 3×3, 5×5) work together inside a single layer instead of using standard convolutional layers. The model uses this feature to detect fine details and large structures at the same time, which proves vital for examining rice leaf images that show various disease signs and patterns and lesions that have different shapes. The model achieves better results because we used an InceptionV3 backbone that had been pretrained on the ImageNet dataset. ImageNet contains more than 1.2 million labeled images across 1,000 categories. The dataset may not contain rice leaf disease information, but it enables the model to acquire basic visual skills that include edge detection and texture recognition and shape identification and color understanding. The learned features enable the model to achieve fast convergence and reduced training time while maintaining high accuracy with minimal local data from each federated client. The team selected InceptionV3 because it delivers outstanding performance while also working well with lightweight optimization methods that include pruning, which serve as vital components for federated learning systems.

Disease Classification Needs Customization

InceptionV3 provides strong feature extraction capabilities but its complete connection system links to 1000 ImageNet classes which do not match our nine rice leaf disease classification system. The model's top classification components were deleted and we built a new lightweight classification system which detects agricultural diseases. The modified architecture includes:

GAP Layer handles multiple feature maps through its average output calculation which helps control model parameters better than traditional full connection systems. The model achieves better performance because of this method which protects against overfitting and enables operation on devices with limited resources.

Dropout Layer (rate = 0.5): To improve generalization, a dropout layer randomly deactivates 50% of neurons during training. The model needs to avoid learning noise because federated clients conduct training with limited data.

The final dense layer uses nine neurons corresponding to the dataset’s nine disease classes. The system uses softmax activation to determine class probabilities.

The researchers used InceptionV3's ImageNet learned representations by freezing about 100 initial training rounds. The custom classification layer established stability after which the team executed unfreezing to achieve their goal. This process enabled the higher-level convolution blocks to develop specific adaptations for rice disease identification through its spotting patterns and lesion boundary and discoloration identification features.

The training method which combines two approaches leads to improved results while maintaining moderate resource consumption.

Pruning with TensorFlow Model Optimization Toolkit (TF- MOT)

The researchers used TFMOT to implement pruning on the dense layers of their model which resulted in improved performance for federated environments. The process of pruning eliminates low magnitude weights from the model which results in smaller model size because these weights have minor impact on output.

The research team established their pruning system through the following configuration:

The researchers defined their polynomial decay schedule through the following equation:

$s(t) = sinitial + (sfinal - sinitial) \cdot (t - tbegin / tend - tbegin)^2$

Where:

• s(t) = Sparsity at training step t

• sinitial = Starting sparsity

• sfinal = Target sparsity

• tbegin, tend = Epochs where pruning starts and ends

The equation describes the relationship between training step t and the current sparsity level which starts at sinitial and reaches sfinal after an established time period.

The pruning schedule establishes a process which gradually decreases weight strength throughout training while maintaining system performance.

Benefits of Optimization

The optimization approach adopted in this project provides several significant advantages, especially for federated learning. The Model Size Reduction helps to lower active parameters through Pruning which achieves a 40% decrease in operational parameters. The model achieves lightweight status because it can operate on mobile devices and low-end equipment that farmers use in their work. The system experiences faster processing times because it requires fewer parameters for both training and inference operations. The system functions better because federated clients need to operate on devices with restricted CPU capabilities which forces them to minimize their computational resources. The communication expenses decrease because smaller models produce less communication costs during federated rounds. The system operates effectively in rural areas with low bandwidth because clients transmit only compressed weight updates. The final accuracy of approximately 98.5% shows that aggressive pruning maintains accuracy at levels which match or exceed the unpruned model. The optimization techniques achieve their purpose because they create an effective combination between system efficiency and performance capabilities.

Federated Learning Setup

The project uses federated learning (FL) as its training method because this system enables different devices and organizations to work together to build a machine learning model without sharing their data. Centralized systems require all data to be sent to one server for training, which creates problems for privacy and security and compliance with regulations. Federated learning enables client devices to train their private data models locally while transmitting only model updates to a central server. The core principles that guide this paradigm include the preservation of data privacy, decentralized learning across distributed clients, and the aggregation of locally trained model parameters on a central server. The system developed by these mechanisms enables organizations to keep their protected data which remains on-site while they build a strong worldwide model. The project adds Lightweight Federated Learning (LWFL) components to enhance system performance in environments with restricted computing power and communication capabilities. Federated learning enables decentralized training processes; however, edge devices still face challenges because the associated models require extensive computational resources, particularly in agricultural settings which rely on basic hardware systems. LWFL solves these problems by implementing methods that decrease the complexity of models while eliminating unneeded layers and parameter compression and core network elements training. The study implements the techniques on a frozen InceptionV3 model which includes several layers that will be kept intact and operates with reduced parameters through weight pruning and decreased server-client communication requirements. The proposed federated system achieves its highest efficiency through these enhancements which enable operation on edge devices without losing detection accuracy. The Flower (FLWR) framework serves as the foundation for the federated architecture which enables researchers to establish federated learning systems through its open-source platform that contains a user-friendly deployment interface. Flower provides complete compatibility with major deep learning frameworks TensorFlow and PyTorch which enables researchers to create both simulated and actual federated testing environments. The client server system of the application enables easy distribution of models

while it handles parameter exchange and aggregation processes which can be modified to decrease communication needs and optimize model training. The features of Flower make it an excellent choice for research which needs to test multiple remote users who work in agriculture using their low power edge devices. The Federated Averaging (FedAvg) algorithm coordinates training activities in the federated system which serves as the main aggregation method used in federated learning workflows. In FedAvg, the server starts the process by sending the global model to all clients who then train the model on their individual datasets during a specified number of training sessions. Clients transmit their modified model weights to the server after finishing their training instead of delivering the actual training data. The server creates an enhanced global model by averaging the weights, which it sends back to clients for their upcoming training session. The project creates five simulated clients, which receive different dataset sections that contain images from all nine classes. The training process at every client starts with all images being resized to 128×128 resolution and then proceeds to execute random flips and rotations for image augmentation while pixels undergo normalization to a 0-1 range and local training continues for five epochs with a 16-image batch size. The client base contains approximately 220 to 280 images per class, which enables students to study both balanced and diverse learning material. The server handles round organization and aggregates updates throughout four global training periods while testing the revised model on a test set that contains 5 percent of the dataset and the system keeps performance logs and checkpoints to enable result restoration and tracking. The research develops and tests the federated learning system, but actual implementations demand better safeguarding methods to ensure complete client data protection and model security. Secure aggregation methods like homomorphic encryption and differential privacy become standard in practical implementations because they stop servers from deducing client details through model updates. The system needs to verify each client who takes part, which prevents unauthorized access, and it must safeguard against operational failures that happen when clients lose connections or send malfunctioning data. Organizations must develop these essential capabilities to function within agricultural networks, which need to expand their operations across various infrastructure environments. Federated learning proves to be highly beneficial for agricultural applications because it enables farmers and institutions to protect their confidential field and crop information which they cannot share due to privacy requirements and limited internet access. Farmers who train models with their local data maintain full control over their data while helping to build a global model which delivers better precision and local accuracy. The local environmental conditions and disease patterns and crop variations of each client model enable it to learn, which helps to build up the global model's overall strength. The federated system permits institutions to collaborate from separate locations while maintaining their data security because it does not need them to gather data in one central location. Agricultural AI systems benefit from federated learning because its combination of privacy protection and flexible design and capacity to expand makes it a powerful solution for their development.

System architecture

The proposed rice leaf disease prediction system uses Lightweight Federated Learning (LWFL) to create its entire architectural design. The system operates as a client server system which enables clients to develop their models through local training on their specific data while the central server manages training by collecting model weight information.

High-Level Architecture Overview

The proposed rice leaf disease prediction system uses a Lightweight Federated Learning (LWFL) framework to enable collaborative model development while keeping agricultural data ownership and privacy rights intact. The system allows training through multiple clients who each possess their own collection of rice leaf disease images which they obtained from field research and local storage facilities. The central server manages decentralized learning operations through two main activities which include sending the global initial model to all clients and gathering their updated models. The system achieves growth in its collective knowledge through every training session because it keeps each device's data collection strictly within its own boundaries. The communication layer which connects clients to the server system enables secure and reliable transmission of model parameters with high operational efficiency. The entire architectural design of this system serves to address both computational requirements and privacy needs which emerge from current precision agriculture systems that collect data from numerous remote locations.

Client-Side Architecture

Federated learning systems operate with their client systems which act as self-contained training units that possess their own storage space and processing power and data preparation abilities. The client holds a confidential collection of rice leaf disease images that show local environmental conditions which include different lighting conditions and various stages of disease advancement and leaf texture variations. The client carries out image preprocessing by resizing all images to a standardized size and performing pixel value normalization while using augmentation methods that include turning the images and flipping them. The model development process uses operations which boost its ability to generalize whereas the local dataset becomes more similar to actual agricultural field conditions. The client obtains the global model which uses a pruned and optimized InceptionV3 backbone after the preprocessing step finishes. The client runs local training for several epochs during which it modifies particular parts of the lightweight system to save processing power. The client demonstrates better efficiency by implementing pruning methods together with selective layer training methods which decrease the count of active system parameters. The system maintains operational stability because of this capability which enables it to function efficiently on devices that have restricted storage space and processing power. The client converts its modified weights into a serialized format which it sends back to the server after finishing the training process. The system operates without transmitting any unprocessed visual data or confidential agricultural information to the server. The client components which include the image loader and preprocessing module as well as the training engine and weight serializer create a secure training environment which maintains its ability to function across different device types and network conditions.

Server-side Architecture

The server operates as the main control system for the federated learning network because it connects all client work to create a single global model. The server starts its training process by creating the base model through a modified InceptionV3 architecture which has been designed for lightweight federated deployment. The server then shares the model parameters with all client participants. The server starts the aggregation process when clients finish their local training and send back their updated weights using the Federated Averaging (FedAvg) algorithm. This method computes a weighted average of the incoming models, ensuring that clients with larger datasets have a proportionally greater influence on the global model. Through this aggregation process the system successfully gathers different disease patterns which exist across various geographic regions while protecting data confidentiality. The server performs model evaluation at every training round by testing the global model with a test dataset which central servers maintain and clients cannot access. This evaluation step provides insights into the model's generalization performance and monitors improvements over time. The server records all activities in detail while it maintains model checkpoints and establishes client connections through an interface that delivers reliable synchronization. The repository contains various model versions which enable users to either restore previous versions or analyze different points in time. The server performs essential functions to create a federated learning environment which maintains its strength together with its transparency and its research method integrity.

Data Flow

The system processes data through its circular and repetitive data flow which matches the federated learning method. The server begins its operations by creating the first global model which it sends to all connected clients. The clients start their work by receiving the model which they use to preprocess their private datasets through data augmentation before they train their models and produce new model weights. The connection between the client and server enables the client's model weights to transfer back to the server. The server combines all received updates to create a new global model which it sends back to clients for their upcoming training session. The system repeats its process multiple times which allows the global model to acquire new knowledge from all client-provided distributed datasets. The system maintains complete data security through its operational method because no unprocessed data can be shared throughout the data processing procedure. The system maintains functional operations even when some clients stop working because client dropout does not affect the ongoing aggregation process during a round when certain clients do not send their data updates.

Hardware and Software Stack

The development and assessment of the federated learning system takes place through simulation testing which operates on an Intel Core i5 processor and 8GB of RAM memory as its testing hardware foundation for real-world applications. The software ecosystem supporting the architecture includes Python 3.10 for scripting, TensorFlow 2.17 for model development and training, and the Flower framework for implementing federated workflows. The

TensorFlow Model Optimization Toolkit (TFMOT) enables model optimization through pruning and lightweighting techniques, while Matplotlib and Scikit-learn provide tools for visualization and performance evaluation. The software tools work together to establish a developing environment which enables federated learning applications to operate at multiple scales while maintaining maximum operational efficiency. The system architecture allows organizations to expand their operations by simply integrating additional clients because the system needs no fundamental alterations to its existing server and client infrastructures. The client system enables users to run their devices with different processing power on their devices because they have complete control over their operations. The server supports dynamic participation, meaning clients can join or leave the training process seamlessly based on availability or connectivity. The scaling capability proves especially useful in agricultural environments because devices in various locations will connect to the network at different times throughout the day.

Experimental Setup

The section presents all practical elements needed to execute federated learning tests together with its required dataset division, system setup, training settings, and testing methods. The designers built the system to replicate actual federated environments while maintaining testing consistency and measuring system performance.

Client Simulation and Partitioning

The research team conducted tests which used five simulated clients to reproduce federated learning testing in their laboratory environment. The clients received different dataset portions which maintained class diversity across their respective partitions.

• The system used 5 simulated clients which operated from a single system through its multiple processing units.

• Each client received data which included all 9 disease categories for their data segment.

• The research used 20% of its complete dataset as a centralized test set which served as the primary evaluation method for global testing.

Data Distribution Strategy

• The data was divided into client allocations which provided each client with 250 images for each class.

• Stratified sampling ensured that each class was evenly distributed across clients.

• The central test set was created through a sampling process which selected specific amounts of samples from each class of the entire dataset before any data partitioning took place.

Model Configuration

The model used a modified InceptionV3 architecture which is optimized for mobile and edge computing environments. The base model uses InceptionV3 which was pre-trained on the ImageNet dataset. The trainable layers contain only the top layers while all bottom layers remain frozen to maintain their learned features. The additional layers consist of GlobalAveragePooling2D and Dropout with 0.5 and Dense which serves as the output layer that uses softmax activation. TFMOT used a polynomial decay schedule to implement the pruning strategy. The system produces 9-class classification results which use softmax for output.

Data Augmentation and Normalization

The system uses two augmentation techniques which include random horizontal flipping and random 15-degree rotation. The system normalization process uses Rescaling at 1./255 to convert pixel values into the [0, 1] range.

Federated Training Configuration

The system will execute three training rounds. The system will execute five training sessions for each client. The system will process data using a batch size of 32. The system uses Adam as its optimization algorithm. The loss function for the system operates using Sparse Categorical Crossentropy. The evaluation process uses Accuracy and Precision and Recall and F1-Score as its measurement standards.

Server Logic and Strategy

The server component used the FedAvg (Federated Averaging) strategy which Flower provides with its own special metric aggregation and logging system.

• Model Aggregation: Weighted average of model parameters based on number of samples per client.

• Evaluation: Central test dataset used for performance validation after each round.

• Metrics Logging: Classification report and confusion matrix printed at the end of training.

Execution Pipeline

1. The server process must be launched with the Flower framework.

2. The system should start five client processes which will work simultaneously.

3. The client systems will conduct local model training and then send weight information back to the central server.

4. The server system combines weight information to create an updated global model.

5. The process continues through three federated rounds of operation.

6. The final model will undergo assessment using the centralized test set.

Monitoring and Logging Tools

• The system uses Matplotlib to display training and validation accuracy data in real-time.

• The system records classification metrics after each round of testing.

• The system stores confusion matrix data and the performance of each class into a file.

Challenges Encountered

• GPU Access: Some clients defaulted to CPU-only execution due to memory overload.
• Synchronization Issues: The system required specific delay times and Flower settings which enabled the system to avoid client server deadlocks.

• Model Size: The system needed pruning to achieve memory space efficiency.

Results and Evaluation

The section displays all results from federated training together with precise metrics which explain how models performed on the central test dataset.

Classification Metrics Overview

The Lightweight Federated Learning system after completing five training sessions which involved training across multiple remote client systems. The assessment of the ultimate global model performance used a test dataset which had been stored in a central location and which clients had never seen during their training sessions. The model evaluation process establishes an accurate assessment of the model's actual ability to generalize to new data.

The model achieved strong convergence results after conducting five federated training sessions which included five local training sessions for each client. The researchers used transfer learning with InceptionV3 to speed up training processes while model pruning maintained system efficiency and resulted in accurate performance outcomes.

Final Results Summary

• Accuracy: 98.5%

• Macro F1-score: 0.9853

• Weighted Precision: 0.9855

• Weighted Recall: 0.9851

Class-wise Report

	precision	recall	f1-score	support
Bacterial_Leaf_Blight	0.96	0.99	0.98	239
Brown_Spot	0.99	0.98	0.99	309
Healthy_Rice_Leaf	1.00	0.99	0.99	217
Leaf_Blast	0.97	0.98	0.98	349
Leaf_scald	0.99	0.97	0.98	266
Narrow_Brown_Leaf_Spot	0.97	0.96	0.97	190
Neck_Blast	1.00	1.00	1.00	200
Rice_Hispa	1.00	1.00	1.00	259
Sheath_Blight	0.99	0.99	0.99	325
accuracy			0.99	2354
macro avg	0.99	0.99	0.99	2354
weighted avg	0.99	0.99	0.99	2354

Key Insights

- Pruning Impact: The assessment showed that pruned layers functioned at the same level as complete models while decreasing their overall size.
- Client Robustness: Clients showed strong performance because they maintained good results despite facing small data imbalance problems.
- Overall Performance: The system reached a point which combined operational efficiency with high-quality prediction results.

Discussion

The project results demonstrate that federated learning together with deep learning methods can successfully work in agricultural research. The proposed solution protects sensitive information on personal devices because it prevents data from being shared with remote servers unlike conventional centralized systems.

The system achieves its main advantage through its capacity to process distributed data while preserving its precise results. This feature proves valuable in agricultural settings because farm data exists in multiple locations throughout different geographical areas and various weather patterns. The model gains increased flexibility through its design which enables client data-based training while maintaining its capacity to handle different real-world situations.

The system benefits from InceptionV3 transfer learning because this method enables operation without requiring extensive training datasets. The model uses its pre-existing learned attributes for training because it needs to operate in situations where there is no access to labelled data.

The system gains improved performance through model pruning because it decreases the amount of processing power needed to operate. This requirement becomes essential for devices that operate in edge environments which include smartphones and the low-power equipment that people use in rural locations.

The research identifies multiple problems which need to be resolved according to its findings. The system depends on all clients to provide dependable participation during every training session which does not match actual operational conditions. The model shows good performance across multiple areas except for its difficulties in recognizing diseases that share visual similarities.

The integration of federated learning together with deep learning and optimization methods creates an effective system for developing AI solutions in agriculture that maintain user privacy and enable large-scale operation.

Future Work

1. Real-World Deployment

Farmers can use the system to take leaf pictures and get immediate results through its installation on both mobile devices and edge computing systems.

2. Larger and Diverse Dataset

Researchers should work on creating a complete dataset that includes:

- Different lighting conditions
- Various camera qualities
- Real-world environmental noise

This will improve the robustness and generalization of the model.

3. Handling Client Heterogeneity

Federated systems need to support multiple client types who use:

- Different hardware capabilities
- Unequal data distributions
- Variable network connectivity

Adaptive training methods will be the focus of future studies which aim to tackle this problem.

4. Communication Optimization

Federated learning requires efficient communication systems because they directly impact system performance. The system can achieve better efficiency through the implementation of:

- Gradient compression
- Model quantization
- Sparse updates

5. Advanced Model Architectures

The use of lightweight architecture developments needs to be investigated through research on:

- MobileNetV2
- EfficientNet
- Vision Transformers

because these advancements will enhance system performance.

6. Security Enhancements

The system requires production-ready status which will be achieved through implementation of advanced privacy techniques:

- Differential Privacy
- Secure Aggregation
- Encryption-based communication

Conclusion

The project achieved its goals by developing and implementing a system which used Lightweight Federated Learning (LWFL) to predict rice leaf diseases with accurate results while safeguarding user privacy. The proposed framework achieved its goals by distributing training to multiple clients while maintaining raw image data on individual devices which solved the key drawbacks found in traditional centralized deep-learning models that

used deep-learning models which required stable internet access and faced data privacy risks. The Flower FL framework established client-server communication connections which enabled local model updates to be combined and training to continue through non-IID (non-uniform) data conditions. The research team used InceptionV3 CNN model which had been pre-trained to achieve better computing efficiency through pruning techniques and data augmentation and normalization methods which cut trainable parameters without damaging model efficiency. The testing of the centrally reserved test dataset demonstrated outstanding classification accuracy while maintaining high precision and recall and F1-scores across all nine rice leaf disease categories. The confusion matrix analysis confirmed model reliability through its ability to distinguish between visually similar disease classes. The system operational success demonstrates how federated learning can benefit agricultural systems which operate in rural regions that face challenges with data security and resource limitations and restricted internet access. The project establishes a solid base for AIdriven plant disease detection tools to operate in real-world scenarios because it allows on-device training and protects user data. The project shows that Lightweight Federated Learning enables the development of smart farming systems which operate efficiently and securely while maintaining their ability to expand. The system will help farmers achieve better results while decreasing crop losses and adopting environmentally friendly farming methods.

References

[1] McMahan, H. B., Moore, E., Ramage, D., et al. (2017). Communication-efficient learning of deep networks from decentralized data. *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 1273–1282.

[2] Li, T., Sahu, A. K., Talwalkar, A., Smith, V. (2020). Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 37(3), 50–60.

[3] Kairouz, P., McMahan, H. B., et al. (2021). Advances and open problems in federated learning. *Foundations and Trends in Machine Learning*, 14(1–2), 1–210.

[4] Howard, A. G., Zhu, M., Chen, B., et al. (2017). MobileNets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*.

[5] Han, S., Mao, H., Dally, W. J. (2015). Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding. *arXiv preprint arXiv:1510.00149*.

[6] Reddi, S., Charles, Z., Zaheer, M., et al. (2021). Adaptive federated optimization. *Proceedings of the International Conference on Learning Representations (ICLR)*.

[7] Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1251–1258.

[8] Kingma, D. P., Ba, J. (2015). Adam: A method for stochastic optimization. *International Conference on Learning Representations (ICLR)*.