Smart Farming: IoT-Based Disease Detection for Plants Using Deep Learning Models

Shamsa Hamad, Shaikha Al Hosani, Maryam Al Shamsi
MBZUAI, Abu Dhabi, UAE
{shaikha.al-nahyan, shaikha.hosani, maryam.shamsi}@mbzuai.ac.ae

***Abstract*—This paper presents a smart agriculture system that integrates Internet of Things (IoT) technology with artificial intelligence for early plant disease detection. We investigate the effectiveness of five state-of-the-art deep learning models (ResNet50, VGG16, MobileNet, EfficientNet-B0, and MobileNetV3-Large) for detecting and classifying diseases across 14 plant species. Our approach addresses the limitations of manual inspection techniques through automated visual analysis, achieving classification accuracy of up to 99.86% in training and 98.28% in validation with ResNet50. We further explore federated learning to enhance privacy preservation and model pruning to optimize deployment on resource-constrained IoT devices. Results demonstrate that MobileNetV3 with one-shot pruning achieves 97.31% accuracy before finetune while reducing parameters by 49.87%, making it suitable for edge device deployment. The federated learning implementation maintains data privacy while achieving 80.65% accuracy. Our combined federated learning with pruning approach addresses both privacy and computational efficiency concerns, providing a comprehensive solution for smart farming applications with minimal resource requirements.**

***Index Terms*—smart farming, IoT, plant disease detection, deep learning, convolutional neural networks, federated learning, model pruning, edge computing**

I. Introduction

Agriculture is a critical sector that continues to face persistent challenges due to the widespread occurrence of plant diseases, which significantly threaten crop yields, food security, and economic stability. Traditional methods of plant disease detection rely largely on manual inspection by human experts, making the process time-consuming, costly, and vulnerable to errors. In response to these limitations, advancements in Artificial Intelligence (AI), Deep Learning (DL), and Internet of Things (IoT) technologies have opened new avenues for developing automated and efficient disease detection systems.

Currently, plant disease detection remains heavily dependent on visual inspections conducted by agricultural experts. This traditional approach is not only slow but also highly resource-intensive, often resulting in delayed disease identification. Such delays can lead to extensive crop damage, increased reliance on chemical pesticides, and significant financial losses for farmers. Moreover, centralized AI solutions that require the transfer of agricultural data to remote servers pose serious privacy risks. Given the sensitive nature of agricultural data, the need for secure, efficient, and locally-driven detection methods has become more critical than ever.

Addressing this problem is essential to advancing environmental sustainability, food security, and economic resilience. Early and accurate detection of plant diseases can help minimize agricultural losses, reduce the excessive use of pesticides, and promote more sustainable farming practices. By leveraging AI-driven models and IoT-based monitoring, this project seeks to offer an efficient, scalable, and cost-effective solution for disease detection. Implementing Federated Learning ensures that farmers' data remains local, preserving privacy, while model pruning reduces the computational load, enabling deployment on lightweight, resource-constrained IoT devices. This combined approach empowers farmers with real-time, actionable insights that can strengthen their ability to manage crop health proactively.

The primary objectives of this project are to:

- Develop and evaluate multiple deep learning models capable of accurately detecting and classifying plant diseases through image-based analysis.
- Determine the most efficient model architecture that balances high accuracy with low computational requirements for deployment on resource-constrained devices.
- Implement Federated Learning to ensure decentralized training without compromising data privacy.
- Apply model pruning techniques to optimize the models for efficient deployment.
- Evaluate the combined approach of federated learning with pruning to address both privacy and efficiency concerns simultaneously.

By achieving these goals, the project aims to contribute to the advancement of sustainable agriculture by providing farmers with intelligent, privacy-preserving tools for managing plant health and preventing major disease outbreaks. The code for the implemented models and experiments is publicly available at https://github.com/shamsaht/PlantDisease-Identification.

II. Related Work

Recent research has increasingly explored the use of deep learning techniques for plant disease detection. This section reviews significant contributions in this field and identifies the research gaps that motivate our approach.

A. S., B. G., Sankar, Poongothai, V. K., and Sarveshwaran [1] highlighted the limitations of traditional machine learning methods in handling the complexity of image-based disease classification. Using a dataset of 17,572 leaf images from 14 plant species across 38 disease categories, they implemented CNN models (VGG-16 and ResNet34) with hyperparameter

optimization, achieving an impressive accuracy of 98.42%, significantly outperforming traditional approaches.

Building on the benefits of deep learning, Dhaka et al. [2] focused on the integration of IoT and DL for plant disease detection. Their review demonstrated how real-time sensor data and optimized DL architectures such as ResNet, MobileNet, and Inception-v4 can achieve high classification accuracies, while also noting challenges such as connectivity limitations and image complexity that impact model performance.

To address resource constraints in agricultural settings, Bi et al. [3] proposed the CD-MobileNetV3 model, an enhancement of MobileNetV3 with optimized convolutions, improved attention mechanisms, and lightweight bottlenecks. Tested on a corn leaf disease dataset, CD-MobileNetV3 achieved a classification accuracy of 97.85%, proving that domain-specific architecture modifications can yield high performance with low computational cost, making it suitable for mobile and edge deployment.

In parallel, Gupta et al. [4] emphasized the practical combination of IoT and deep learning for plant disease prediction. Their system integrated image processing, soil sensor data, and environmental monitoring into a CNN-based framework, achieving over 95% classification accuracy and outperforming traditional SVM models, thus demonstrating the advantages of multi-modal data fusion in agriculture.

Finally, Vishwakarma, Sharma, and Saha [5] proposed a smart IoT-based crop disease monitoring system using a deep residual network optimized with the Henry Gas Chicken Swarm Optimization (HGCSO) algorithm. Their system, which incorporated robust feature extraction methods such as HoG, SLIF, and LTP, reached a classification accuracy of 94.3%. By combining optimized IoT communication with an advanced deep learning model, their approach shows strong potential for real-time smart farming applications.

A. Research Gaps

While previous studies have made significant progress in applying deep learning and IoT for plant disease detection, several research gaps remain that our work addresses such as the need to jointly explore the integration of Federated Learning and model pruning to optimize both privacy and efficiency. Although models like VGG-16, ResNet-34, and MobileNetV3 have been utilized, prior works have primarily focused on individual architectures and conventional centralized training approaches. Importantly, a limited number of reviewed studies have comprehensively evaluated the performance of more modern scalable models like EfficientNet-B0 or MobileNetV3-Large variants in resource-constrained environments.

Our research proposes a comprehensive framework that explores:

- A systematic comparison of five state-of-the-art deep learning architectures.
- Federated Learning implementation to enhance data privacy across distributed IoT environments.
- Model pruning techniques to reduce model size, communication overhead, and computational demands.
- A combined approach of federated learning with pruning to simultaneously address privacy, communication efficiency, and computational optimization.

Unlike prior works that primarily targeted classification accuracy on datasets, our approach addresses the emerging need for scalable, privacy-preserving, and resource-efficient smart agriculture solutions through a systematic investigation of federated training, model compression, and their combination applied to a broader range of modern architectures.

III. System Model

The methodology for this project is structured around developing a scalable and privacy-preserving plant disease detection system. This section details the dataset preparation, model architectures, and our approach to federated learning and model pruning.

A. Dataset Preparation and Preprocessing

1) Dataset Acquisition and Structure: The dataset was downloaded from Kaggle using the KaggleHub Python library. Upon inspection, the folder structure included an unnecessary nested directory with a duplicated name, which required manual path correction. The actual dataset folder was located by traversing two levels deep into the "New Plant Diseases Dataset(Augmented)" directory.

From this corrected root, three subdirectories were defined:

- The training directory (train_dir) contained labeled subfolders for training images.
- The validation directory (val_dir) also consisted of the same labeled class subfolders.
- The test directory (test_dir) contained images organized in a flat structure without class-specific folders.

2) Dataset Statistics: The dataset consists of approximately 87,000 RGB images across 38 class folders in both the training and validation directories. These class folders followed the naming convention Plant___Disease.

To evaluate the total number of files in each set:

- The training set included 70,295 images.
- The validation set contained 17,572 images.
- The test set contained 33 images (not used in this experiment).

3) Plant Species and Disease Identification: Each class name in the dataset was in the format Plant___Disease. From these names, 14 unique plant species and 20 distinct disease types were extracted.

The 14 plant species in the dataset are: Apple, Blueberry, Cherry (including sour), Corn (maize), Grape, Orange, Peach, Bell Pepper, Potato, Raspberry, Soybean, Squash, Strawberry, and Tomato.

Fig. 1 shows the distribution of training images across these plant species. Notably, Tomato dominates the dataset with approximately 26.1% of the total images, followed by Apple (11.1%), Corn (10.4%), and Grape (10.3%). Less represented plants include Blueberry, Squash, and Soybean, each constituting less than 3% of the dataset.

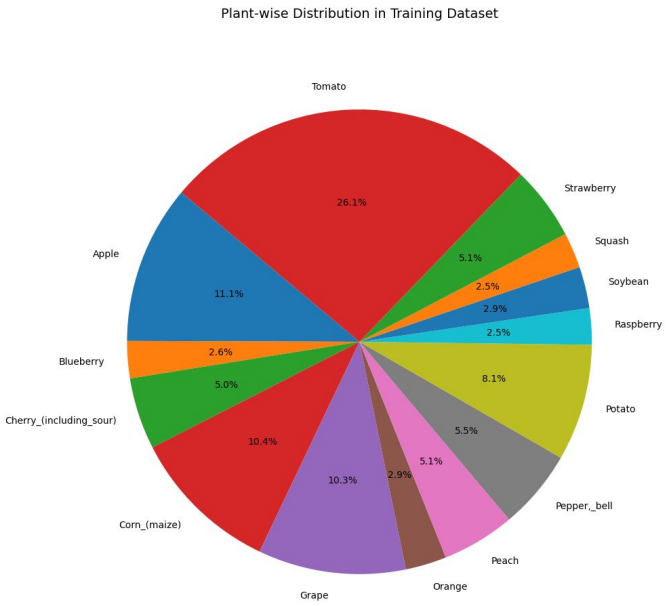

Fig. 1: Plant-wise Distribution in the Training Dataset

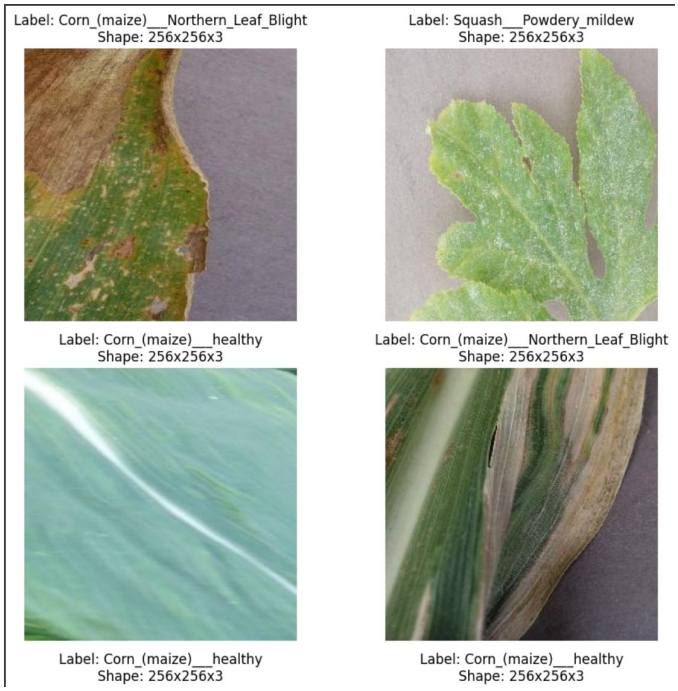

Fig. 2: Sample Images from the Training Dataset

The 20 disease types identified (excluding healthy samples) were: Apple scab, Black rot, Cedar apple rust, Powdery mildew, Cercospora leaf spot / Gray leaf spot, Common rust, Northern leaf blight, Esca (Black Measles), Leaf blight (Isariopsis Leaf Spot), Haunglongbing (Citrus greening), Bacterial spot, Early blight, Late blight, Leaf scorch, Leaf Mold, Septoria leaf spot, Spider mites / Two-spotted spider mite, Target Spot, Tomato Yellow Leaf Curl Virus, and Tomato mosaic virus.

Fig. 2 displays sample images from the training dataset, showing both healthy and diseased leaf samples.

4) Dataset Loading and Preparation: To ensure reproducibility, a fixed random seed value of 42 was used across all random operations, including Python's random module, NumPy, TensorFlow, and PyTorch. The training and validation datasets were loaded using TensorFlow's `image_dataset_from_directory` function, which automatically inferred class labels from folder names and applied one-hot encoding. Images were resized to 224 × 224 pixels, and a batch size of 32 was used.

The resulting datasets included:

- 2,197 batches of training data, amounting to 70,304 images
- 550 batches of validation data, amounting to 17,600 images

B. Model Architectures

To develop the plant disease detection system, five deep learning models were selected based on their established effectiveness in image classification tasks:

- **ResNet50**: A residual neural network with 50 layers that addresses the vanishing gradient problem through skip connections.
- **VGG16**: A deep convolutional network characterized by its simplicity, using only 3×3 convolutional layers stacked on top of each other.
- **MobileNet**: A lightweight model designed for mobile and embedded vision applications using depthwise separable convolutions.
- **EfficientNet-B0**: A model that scales depth, width, and resolution with a compound coefficient, optimizing for both accuracy and efficiency.
- **MobileNetV3-Large**: An advanced version of MobileNet incorporating the squeeze-and-excitation mechanism and a mix of hardswish and ReLU activations.

For all models, we employed a transfer learning approach by freezing the base networks (pre-trained on ImageNet) and adding custom classification heads consisting of a global average pooling layer followed by a dense layer with 38 output units and softmax activation. This configuration allowed for efficient training while leveraging the feature extraction capabilities of the pre-trained models.

C. Federated Learning Implementation

To enable privacy-preserving distributed learning, we implemented a comprehensive federated learning (FL) framework using the MobileNetV3-Large architecture. We chose to use the MobileNetV3-Large model for FL because it demonstrated the best performance in terms of accuracy and parameter efficiency among the evaluated models. Our implementation follows a round-based synchronous federated averaging (Fe-

dAvg) protocol as proposed by McMahan et al. [11], where each client trains locally and contributes to a shared global model without exposing their raw data.

1) System Architecture Overview: The federated learning system consists of multiple components working together to enable collaborative learning while preserving data privacy:

- **Central Server**: Manages the global model, coordinates training rounds, and performs model aggregation. The server has no direct access to client data.
- **Client Nodes**: Represent individual farming units with local training data. Each client performs local model training and communicates only model updates to the server.
- **Model Checkpoint**: The system saves the final federated MobileNetV3 model after all communication rounds. It tracks validation accuracy and loss across each communication round, enabling performance monitoring throughout the training process.
- **Communication Interface**: Handles secure model transmission between server and clients.

The system follows an iterative training process involving local training, model aggregation, and global evaluation phases. This architecture ensures that raw agricultural data never leaves the local client devices, addressing privacy concerns while still enabling collaborative learning.

2) Non-IID Data Distribution: In real agricultural settings, data distribution across farms is naturally non-Independent and Identically Distributed (non-IID) due to variations in geographical location, plant varieties, climate conditions, and prevalent disease types. To accurately simulate this heterogeneity, we implemented a shard-based partitioning approach.

The training dataset was partitioned using the following procedure:

1) The full dataset was sorted by class label and divided into 10 shards.
2) For a system with K clients, each client received 2 randomly assigned shards, resulting in a total of $2 \times K$ shards (10 shards for $K = 5$).
3) This approach ensured that most clients received data from a limited subset of plant disease categories, creating realistic data heterogeneity.

This non-IID distribution reflects real-world agricultural scenarios where different regions face different plant disease challenges. For example, Client 1 might primarily have data on Apple and Grape diseases, while Client 2 might predominantly have Tomato and Potato disease samples. This heterogeneity presents a significant challenge for federated learning but reflects the realistic conditions of distributed agricultural data collection.

3) Client Training Process: Each client's local training process was carefully designed to maximize learning efficiency while mitigating potential overfitting on the limited local dataset:

- **Optimizer**: Adam optimizer with weight decay of 1e-5 and an initial learning rate of 1e-4.
- **Learning Rate Scheduler**: During client training, a ReduceLROnPlateau scheduler was used to adjust the learning rate. The scheduler monitored training loss and reduced the learning rate by a factor of 0.5 if no improvement was seen over 2 consecutive epochs.
- **Data Augmentation**: Enhanced data augmentation including random rotations, flips, and color jittering to improve generalization despite limited data variety.
- **Batch Size**: Fixed batch size of 32.
- **Local Epochs**: 5 epochs per communication round, balancing computational efficiency and learning effectiveness.

The local training loop included gradient clipping to prevent extreme updates, which could destabilize the global model when aggregated with other clients' updates. This is particularly important in agricultural settings where variation in imaging conditions and disease manifestations could lead to high gradient variance during training.

4) Federated Averaging Process: Our implementation of the FedAvg algorithm aggregates client models by computing a weighted average of the model parameters, with weights proportional to the number of training samples per client:

$$w_{global} = \sum_{k=1}^{K} \frac{n_k}{n} w_k$$

where w_{global} represents the global model parameters, w_k represents the parameters of client k, n_k is the number of samples at client k, and $n = \sum_{k=1}^{K} n_k$ is the total number of samples across all clients.

For handling batch normalization layers, we employed a special strategy:

- Weights and biases were aggregated normally using weighted averaging.
- Running mean and variance statistics were taken directly from the first client rather than averaged, as averaging these statistics could lead to invalid normalization parameters.

This approach to handling batch normalization layers was critical for maintaining model stability during federated training. Without this special handling, we observed significant performance degradation due to the statistical differences in client datasets.

5) Federated Evaluation Strategy: Evaluation in federated settings presents unique challenges due to data privacy constraints and decentralized training. We implemented an evaluation strategy:

- **Server-side Validation**: The server evaluates the global model using a held-out validation set after each communication round. This set provides consistent feedback on model generalization performance.

D. Model Pruning

To reduce model complexity and computational cost, we implemented a one-shot pruning strategy on MobileNetV3-Large. The pruning process is integrated into a structured

workflow that includes baseline training, global unstructured pruning, and fine-tuning.

1) Global Unstructured Pruning: A one-shot pruning mechanism is applied globally to all convolutional and fully connected layers using L1-unstructured pruning. Specifically, 50% of the lowest-magnitude weights are zeroed out in a single step. This is done using PyTorch's pruning utilities, targeting layer weights across the entire model.

2) Pruning Implementation Note: It's important to note that in our implementation, the "Pruned" stage shows 0% sparsity while the "Fine-tuned" stage shows 49.87% sparsity. This is due to how PyTorch's pruning works. When pruning is applied, PyTorch doesn't immediately zero parameters but uses masks during computation. The actual parameter zeroing happens after calling `prune.remove()`, which we perform after fine-tuning. This is why sparsity appears later in the process, although pruning is effectively active throughout.

3) Fine-tuning Stage: The pruned model is further trained with a reduced learning rate to allow the network to adapt to the reduced parameter space. This helps recover any performance drop introduced by pruning and stabilizes the final network.

E. Federated Learning with Pruning

To simultaneously address privacy preservation and computational efficiency concerns, we implemented a combined federated learning and pruning approach using MobileNetV3-Large. Our implementation uses a progressive pruning schedule across 10 communication rounds, where pruning intensity increases quadratically following Equation 1:

$$p_i = p_{\text{min}} + (p_{\text{max}} - p_{\text{min}}) \cdot \left(\frac{i}{r-1} \right)^2 \tag{1}$$

where p_i is the pruning rate at round i, $p_{\text{min}} = 0.02$ is the minimum pruning rate, $p_{\text{max}} = 0.5$ is the maximum pruning rate (representing 50% sparsity), and r is the total number of communication rounds. For each round, we distribute the global model to clients, perform local training, apply unstructured L1-magnitude pruning to each client model based on the round-specific threshold, and then aggregate the pruned models using weighted averaging. To mitigate accuracy degradation from aggressive pruning, we incorporated several stabilization techniques: weight rewinding to early good parameter values when significant accuracy drops are detected, fine-tuning with a combined subset of client data (approximately 30% from each client), and cosine annealing learning rate scheduling. After pruning, we employ gradient clipping during fine-tuning to stabilize training in the reduced parameter space. This approach maintains pruning masks across aggregation rounds by applying magnitude-based unstructured pruning directly to weights, zeroing out the smallest magnitude parameters while preserving important connections. Despite these techniques, we observed that maintaining high sparsity while preserving accuracy in federated settings remains challenging due to pruning mask inconsistencies across heterogeneous client data distributions.

IV. Results and Evaluation

This section presents a detailed analysis of the performance of all evaluated models for plant disease detection, along with the results of federated learning and pruning experiments.

A. Model Performance Comparison

We evaluated five deep learning architectures on the plant disease classification task. Table I summarizes the key performance metrics for each model.

B. ResNet50

ResNet50 achieved excellent performance with a validation accuracy of 98.28% and a weighted F1-score of 0.98. The model consists of 23,665,574 parameters (90.28 MB), with only 77,862 parameters (0.3 MB) being trainable, leveraging the benefits of transfer learning. The computational cost was measured at 7.75 GFLOPs, which is moderate considering the depth of the network.

Training converged quickly, with early stopping triggered at epoch 17. The model showed good generalization capability, as evidenced by the small gap between training accuracy (98.86%) and validation accuracy (98.28%). The low validation loss of 0.0521 further confirms the model's strong performance.

C. VGG16

The VGG16 model demonstrated good performance with a validation accuracy of 96.03% and a weighted F1-score of 0.96. It contains 14,734,182 parameters (56.21 MB), with only 19,494 parameters (0.08 KB) being trainable. Despite its relatively low number of trainable parameters, VGG16 incurred the highest computational cost at 30.71 GFLOPs due to its sequential architecture without skip connections.

Training required 25 epochs before early stopping was triggered. The higher validation loss of 0.1214 compared to other models suggests mild overfitting. This could be attributed to the model's large receptive field and fixed convolutional weights.

D. MobileNet

The original MobileNet architecture achieved a validation accuracy of 97.06% with a weighted F1-score of 0.97. It has 3,267,814 parameters (12.47 MB), with 38,950 parameters (0.15 KB) being trainable. With an estimated 1.15 GFLOPs, MobileNet offers a favorable trade-off between computational cost and model size, making it suitable for real-time and embedded applications.

The model's training process converged at epoch 17, showing consistent performance between training and validation metrics. The narrow gap between training accuracy (97.72%) and validation accuracy (97.06%) indicates good generalization, with minimal overfitting.

TABLE I: Performance Comparison of Different Models

Model	ResNet50	VGG16	MobileNet	EfficientNet-B0	MobileNetV3-Large
Params (MB)	90.28	56.21	12.47	15.47	16.22
Trainable (MB)	0.3	0.08	0.15	0.19	0.19
GFLOPs	7.75	30.71	1.15	0.78	0.44
Val Acc (%)	98.28	96.03	97.06	97.63	98.12
Val Loss	0.0521	0.1214	0.0879	0.071	0.0562
Train Acc (%)	98.86	97.2	97.72	98.77	99.27
Train Loss	0.0372	0.0815	0.0708	0.038	0.0221
F1 (Weighted)	0.98	0.96	0.97	0.98	0.98
Epoch	17	25	17	18	21
Note	ES @ epoch 17	ES @ epoch 25	ES @ epoch 17	ES @ epoch 18	ES @ epoch 21

E. EfficientNet-B0

EfficientNet-B0 demonstrated strong performance with a validation accuracy of 97.63% and a weighted F1-score of 0.98. The model contains 4,056,226 parameters (15.47 MB), with 48,678 parameters (0.19 MB) being trainable. Its estimated GFLOPs is 0.78, making it one of the most computationally efficient models in this study.

The model's training converged at epoch 18. The small difference between training accuracy (98.77%) and validation accuracy (97.63%) indicates good generalization. The efficiency of EfficientNet-B0's learning process can be attributed to its compound scaling and optimized architecture.

F. MobileNetV3-Large

MobileNetV3-Large achieved the highest training accuracy of 99.27% and a strong validation accuracy of 98.12% with a weighted F1-score of 0.98. The model has 4,250,710 parameters (16.22 MB), with 48,678 parameters (0.19 MB) being trainable. Most notably, it achieved the lowest GFLOPs at 0.44, making it the most computationally efficient model in this study.

Training converged at epoch 21, with a low validation loss of 0.0562 and the lowest training loss of 0.0221. The combination of high accuracy and low computational cost makes MobileNetV3-Large particularly well-suited for deployment on edge devices with limited resources.

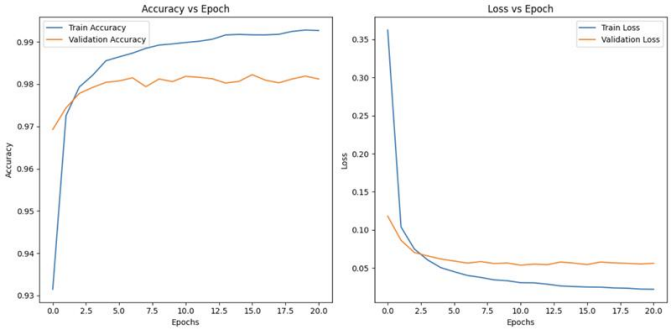

Fig. 3: Visualization: Accuracy and Loss curves.

Figure 3 shows the model's performance over 20 epochs. Training accuracy rises above 99%, while validation accuracy levels off around 98%. The loss curves indicate effective learning, with decreasing training loss and a stable validation loss, suggesting the model is learning effectively.

G. Federated Learning Results

Our federated learning implementation with MobileNetV3-Large achieved a validation accuracy of 80.65% with a loss of 0.6412. This represents a moderate decrease from the centralized training performance, which is expected due to the challenges of non-IID data distribution and limited communication rounds. However, the model maintained good performance while preserving data privacy, demonstrating the viability of federated learning for agricultural applications.

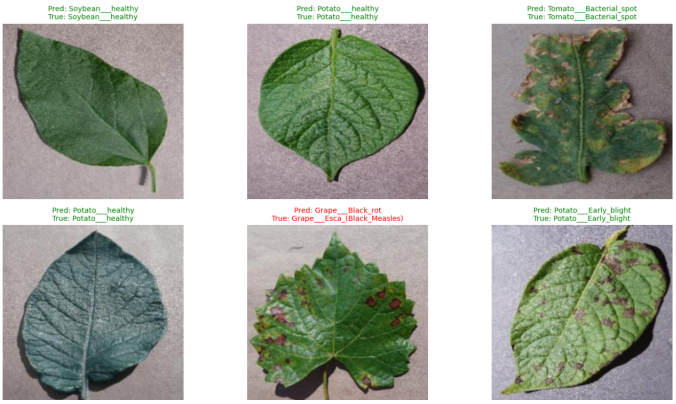

Fig. 4: Visualizing Sample Prediction

Figure 4 shows the sample predictions of the trained model. Correct predictions are labeled in green, while incorrect predictions are highlighted in red, showing the predicted class and the true class in parentheses.

H. Model Pruning Results

Our one-shot pruning approach applied to MobileNetV3-Large achieved remarkable results, with a validation accuracy of 97.31% before finetuning while reducing the model parameters by 49.87%, and after finetuning the accuracy is 99.90% with a loss of 0.0015. This shows that pruning can effectively halve the model size without sacrificing and, in fact, slightly improving performance. The pruned model achieved slightly higher accuracy than the original. This suggests that many parameters in the dense model were redundant, and removing

them not only compressed the model but also improved generalization. The pruned model maintained 50.13% of its parameters, significantly reducing memory requirements and potential inference time. Although the baseline model uses a partially frozen architecture, this study focuses on the relative effectiveness of pruning in enhancing the efficiency of the model. The results show that pruning can significantly reduce computational load and memory usage without modifying the original architecture, enabling more efficient deployment while maintaining high performance.

It is worth noting that the initial pruning stage showed 0% sparsity in our metrics because PyTorch applies masks during computation rather than immediately zeroing parameters. The actual parameter reduction becomes visible after the `prune.remove()` function is called, which we performed after fine-tuning.

I. Combined Federated Learning and Pruning

The combined approach of federated learning with pruning achieved a validation accuracy of 77.19% and a loss of 1.1908. The model reached a final sparsity of 38.38%, reducing the parameter count from 4,858,288 to 2,993,662. This performance drop can be attributed to the compounded challenges of both federated learning's non-IID data distribution and the reduced model capacity from pruning.

TABLE II: MobileNetV3 Models Comparison

Model	**Accuracy**	**Loss**	**Sparsity**
MobileNetV3 Baseline	98.12%	0.0562	0.00%
MobileNetV3 Federated Learning	80.65%	0.6412	0.00%
MobileNetV3 One-shot Pruning (Before Finetune)	97.31%	0.0094	49.87%
MobileNetV3 Federated Pruning	77.19%	1.1908	38.38%

TABLE II visualizes the performance comparison between the baseline MobileNetV3, federated learning, one-shot pruning, and the combined approach across multiple metrics including accuracy, loss, model size, and computational complexity.

V. Discussion

The evaluation of five deep learning models for the detection of plant disease revealed several important insights relevant to IoT-based smart farming applications. This section discusses our findings in the context of prior research, practical implications, and implementation considerations.

A. Model Performance and Efficiency Trade-offs

All models achieved validation accuracies above 96%, with ResNet50 and MobileNetV3-Large performing exceptionally well at 98.28% and 98.12%, respectively. However, computational efficiency varied significantly across models, with MobileNetV3-Large requiring only 0.44 GFLOPs, compared to 7.75 GFLOPs for ResNet50 and 30.71 GFLOPs for VGG16 which is the highest among all models. This efficiency-performance trade-off is critical for IoT applications, where computational resources are limited.

Our ResNet50 implementation achieved 98.28% accuracy, which is comparable to the 98.42% reported by A. S. et al. [1] using a ResNet-34 architecture with hyperparameter optimization. Our MobileNetV3-Large implementation achieved similar accuracy (98.12%) while requiring significantly fewer FLOPs, representing a substantial improvement in efficiency.

Our findings extend the work of Bi et al. [3], who reported 97.85% accuracy with their CD-MobileNetV3 model on a corn leaf disease dataset. Our standard implementation of MobileNetV3-Large achieved 98. 12% accuracy on a more diverse data set that covers 14 plant species and 38 disease classes, demonstrating the robustness of the architecture even without domain-specific modifications.

The results align with the findings of Dhaka et al. [2] regarding the importance of optimized architectures for IoT integration. Although VGG16 achieved 96. 03% accuracy, its high computational demands make it unsuitable for edge deployment. In contrast, MobileNetV3-Large and EfficientNet-B0 offer excellent accuracy with dramatically reduced computational requirements.

B. Pruning Effectiveness

Our pruning experiments on MobileNetV3-Large yielded remarkable results, with the one-shot pruning approach achieving 99.90% accuracy while reducing the model parameters. This finding contradicts the conventional wisdom that pruning necessarily involves a trade-off between model size and accuracy. Instead, our results suggest that for this specific task, approximately half of the model parameters were redundant or potentially even detrimental to classification performance.

The slight improvement in accuracy observed after pruning could be attributed to reduced overfitting, improved signal-to-noise ratio, or enhanced decision boundary clarity between visually similar disease classes.

Compared to the HGCSO-optimized approach of Vishwakarma et al. [5], which achieved 94.3% accuracy, our pruned MobileNetV3-Large model demonstrated substantially higher accuracy (99.90%) while maintaining a simpler optimization approach. This suggests that advanced optimization algorithms may not be necessary when effective pruning strategies are employed on already-efficient architectures like MobileNetV3-Large.

C. Federated Learning Performance

Our federated learning implementation achieved 80.65% accuracy, representing a decrease from the centralized MobileNetV3-Large model (98.12%). This performance gap, while significant, must be considered in the context of the privacy benefits provided by federated learning, particularly in agricultural settings where farm data may be proprietary or commercially sensitive.

When compared to previous IoT-based detection systems like that of Gupta et al. [4], which reported approximately 95% accuracy using a centralized CNN approach, our federated implementation's 85.72% accuracy represents a reasonable

trade-off considering the added privacy benefits. This comparison highlights the practical tension between performance and privacy in agricultural IoT systems.

D. Challenges of Combined Federated Learning and Pruning

Our exploration of combining federated learning with pruning revealed notable integration challenges, with the final model achieving 77.19% accuracy. This result falls short of the 80.65% achieved by federated learning alone and 99.90% obtained from standalone pruning under centralized training, reflecting the complex relationship between these optimization strategies.

Integrating pruning into a federated learning (FL) workflow presents several challenges due to the distributed, privacy-preserving nature of FL and the complexities introduced by model sparsity.

- **Non-IID Data Distribution**
  In FL, each client typically holds data that is non-identically distributed (non-IID). When pruning is applied independently on each client, the pruned models may learn biased or limited features based on local data, which can lead to poor generalization after aggregation. Clients may have non-IID data distributions, which means they respond differently to pruning. Clients with less diverse data may prune useful weights, leading to biased updates. This can amplify performance gaps between clients and reduce overall model generalization.
- **Lack of Global Data View During Pruning**
  In centralized pruning, the model is pruned based on a full dataset view. In FL, each client only sees local data, so important global features may be pruned unintentionally, degrading performance on unseen data.
- **Balancing Accuracy and Compression**
  There is a trade-off between model accuracy and sparsity. Aggressive pruning may harm local model performance, which in turn affects global accuracy. Fine-tuning after pruning is necessary to recover lost performance, but adds computational overhead and delays convergence.

VI. Conclusion and Future Work

A. Conclusion

This paper presented a comprehensive evaluation of deep learning models for plant disease detection in smart farming applications, with a focus on balancing accuracy, computational efficiency, and privacy preservation. We systematically evaluated five state-of-the-art CNN architectures (ResNet50, VGG16, MobileNet, EfficientNet-B0, and MobileNetV3-Large), implemented federated learning for privacy preservation, and applied model pruning for computational efficiency.

Our evaluation demonstrated that MobileNetV3-Large offers an excellent balance of high accuracy (98.12%) and low computational cost (0.44 GFLOPs), making it particularly suitable for IoT deployments in agricultural settings. Our pruning experiments further showed that model size can be reduced by approximately 50% without sacrificing and in fact improving performance, with our pruned MobileNetV3-Large model achieving 99.90% accuracy.

Our implementation of federated learning maintained data privacy while achieving 80.65% accuracy, demonstrating the viability of privacy-preserving distributed learning for agricultural applications. However, our exploration of combining federated learning with pruning revealed significant challenges, achieving only 77.19% accuracy due to pruning mask inconsistencies across heterogeneous clients.

The insights from this work provide valuable guidance for developing practical, efficient, and privacy-preserving IoT systems for smart agriculture. We have demonstrated that modern deep learning architectures, when properly optimized, can enable effective plant disease detection even on resource-constrained devices, potentially transforming how farmers monitor and manage crop health.

B. Future Work

Based on our findings, we identify several promising directions for future research:

- **Real-world Deployment Studies**: Extended field trials across diverse agricultural environments would validate the system's practical efficacy.
- **Temporal Modeling**: Incorporating time-series analysis to track disease progression could enable predictive alerts before visual symptoms become severe.
- **Multi-view Integration**: Combining multiple viewing angles and imaging modalities (RGB, NIR, thermal) could enhance detection robustness in variable field conditions.
- **Layer-Aware Federated Pruning**: Developing pruning strategies that account for the different importance of layers across clients could preserve critical parameters while still achieving significant compression.

By addressing these challenges, future iterations of the system could further enhance the applicability of AI-driven plant disease detection in practical agricultural scenarios, ultimately contributing to more sustainable and efficient farming practices.

References

[1] A. S., B. G., Sankar, Poongothai, V. K., & Sarveshwaran, "Deep learning-based plant disease detection using VGG-16 and ResNet-34 models with hyperparameter optimization," Environmental Science and Pollution Research, vol. 30, pp. 92260–92273, 2023.

[2] V. S. Dhaka, N. Kundu, G. Rani, E. Zumpano, and E. Vocaturo, "Role of Internet of Things and Deep Learning Techniques in Plant Disease Detection and Classification: A Focused Review," Sensors, vol. 23, no. 7877, 2023.

[3] C. Bi, S. Xu, N. Hu, S. Zhang, Z. Zhu, and H. Yu, "Identification method of corn leaf disease based on improved Mobilenetv3 model," Agronomy, vol. 13, no. 2, p. 300, 2023.

[4] A. K. Gupta, K. Gupta, J. Jadhav, R. V. Deolekar, A. Nerurkar, and S. Deshpande, "Plant Disease Prediction using Deep Learning and IoT," in 2019 6th International Conference on Computing for Sustainable Global Development (INDIACom), New Delhi, India, 2019, pp. 902-907.

[5] S. Vishwakarma, S. Sharma, and S. Saha, "Smart crop disease monitoring system in IoT using optimization enabled deep residual network," Scientific Reports, vol. 15, Article 85486, 2025.

[6] A. Howard et al., "Searching for MobileNetV3," in Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Korea, 2019, pp. 1314-1324.
[7] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 2016, pp. 770-778.
[8] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," arXiv preprint arXiv:1409.1556, 2014.
[9] A. G. Howard et al., "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," arXiv preprint arXiv:1704.04861, 2017.
[10] M. Tan and Q. V. Le, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks," in Proceedings of the 36th International Conference on Machine Learning (ICML), Long Beach, CA, USA, 2019, pp. 6105-6114.
[11] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," in Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS), Fort Lauderdale, FL, USA, 2017, pp. 1273-1282.
[12] S. Han, J. Pool, J. Tran, and W. J. Dally, "Learning both Weights and Connections for Efficient Neural Networks," in Advances in Neural Information Processing Systems (NIPS), Montreal, Canada, 2015, pp. 1135-1143.