DOI: 10.5281/zenodo.18015377

A Federated Learning-Based Object Detection System with Edge–Cloud Collaboration

Yunsheng Wang, Xiangning Chen, Shubin Wang*

School of Electronic Information Engineering, Inner Mongolia University
Correspondence Author

Abstract: ***With the development of smart agriculture, field object detection has become a key component in improving management efficiency. This paper proposes and implements a cloud-edge collaborative federated learning system for object detection, using a field weed dataset as a case study. The system employs a federated learning framework to enable parallel training across multiple edge devices and integrates the YOLOv11 model to perform collaborative training on several Jetson Nano devices. To enhance global model performance and convergence speed, only model parameters that meet predefined accuracy thresholds and pass a target filtering mechanism are uploaded to the cloud for aggregation. On the client side, a PyQt5-based graphical user interface is developed to support inference on images, videos, and real-time camera feeds.In the experimental evaluation, we compared federated training across multiple edge devices with centralized training on a single device. The results show that federated learning outperforms centralized training across several key performance metrics. Specifically, precision improved from 77.89% to 82.17%, recall increased from 72.25% to 73.56%, and mean average precision (mAP) rose from 77.82% to 81.45%. These findings demonstrate that federated learning can significantly enhance model accuracy and generalization while keeping data localized.***

Keywords: Federated Learning; Cloud-Edge Collaboration; YOLOv11.

1. INTRODUCTION

In recent years, agriculture worldwide has been progressively shifting toward intelligent and precision-based practices. Traditional labor-intensive management approaches are often inefficient and susceptible to subjective judgment. With the continuous refinement of computer vision and deep learning technologies, object detection algorithms—particularly the YOLO series—have been extensively applied in agricultural production, exhibiting excellent performance in tasks such as crop recognition and detection[1]. Recent studies have integrated YOLO-based object detection algorithms into federated learning frameworks, such as the FedVision platform [2]and the FedDet system[3]. They have conducted preliminary explorations of multi-device collaborative training and object recognition tasks.

However, most existing approaches rely on centralized deep learning frameworks, which require large volumes of field image data to be uploaded to cloud servers for training. Due to limitations such as network bandwidth and communication latency, these methods are often ill-suited for real-world agricultural deployments, where devices are distributed across fields and computational resources are constrained. To address these challenges, this paper adopts federated learning, a technique that enables each edge device to perform model training locally while only uploading model parameters. Federated learning not only ensures data privacy but also significantly reduces communication overhead, and has been widely adopted in data-sensitive domains such as healthcare, finance, and security in recent years[4][5]. To further enhance computational and communication efficiency, an edge–cloud collaborative architecture is introduced into the federated learning framework. In this architecture, edge devices are responsible for local data processing and model training, while the cloud server handles model aggregation and overall coordination[6].

This paper proposes and implements a federated learning-based object detection system with edge–cloud collaboration, specifically designed for farmland weed recognition scenarios. A graphical user interface (GUI) for the client side is developed using PyQt5. The system integrates key technologies such as federated learning, edge–cloud collaboration, and visual inference, offering strong practicality, deployability, and scalability.

2. Related Concepts and Theoretical Foundations

2.1 Cloud-Edge Collaboration

With the growing use of IoT and edge devices, traditional cloud-centric architectures face challenges like latency,

privacy risks, and bandwidth limitations. Cloud-edge collaboration addresses these issues by combining cloud and edge computing strengths. As shown in Figure 1, the cloud handles model aggregation and control, while edge devices perform lightweight inference and local training. This distributed setup keeps data local and enables synchronized model updates, improving efficiency, privacy, and scalability.

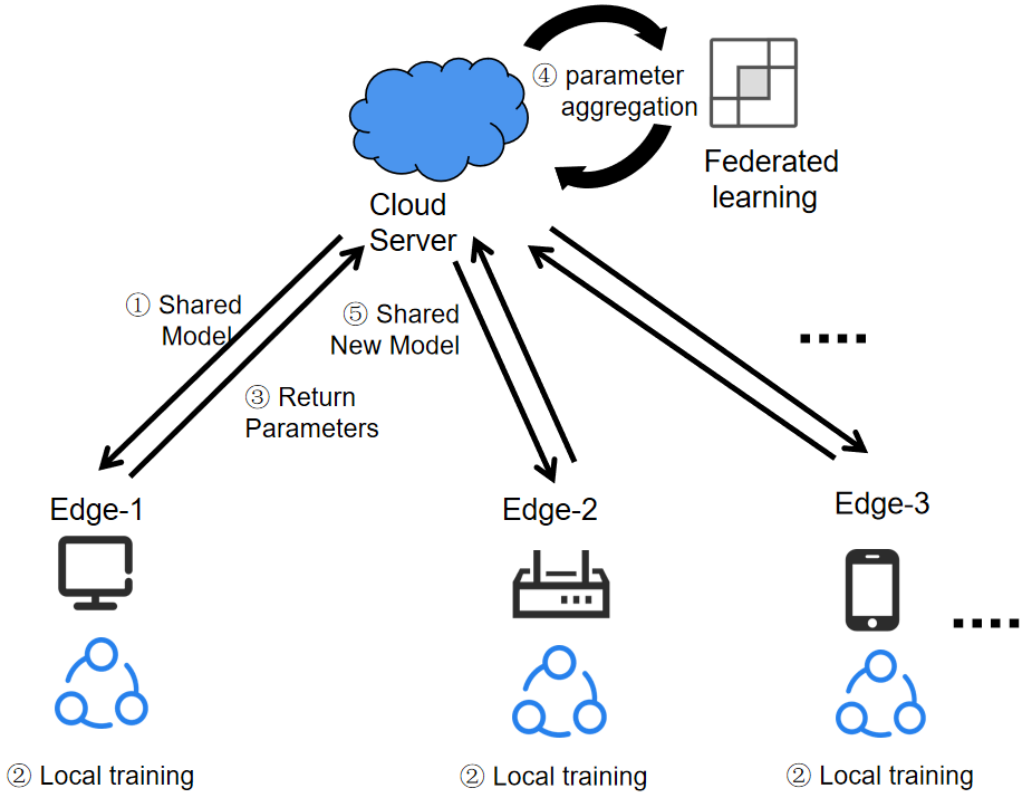

Figure 1: Cloud-Edge Collaboration

2.2 Federated Learning

Traditional centralized deep learning requires uploading data to a central server, which risks privacy breaches in sensitive scenarios. Federated learning offers an alternative by training models across distributed devices without sharing raw data, enhancing privacy while maintaining strong performance.

One of the most well-known aggregation algorithms in federated learning is Federated Averaging (FedAvg). Its core principle is to update the global model by performing a weighted average of the model parameters uploaded by clients, based on the number of local data samples. The formula is as follows:

$$\omega_{t+1} = \sum_{k=1}^{K} \frac{n_k}{n} \omega_t^k$$

Where: ω_{t+1} is the updated global model at round t+1, ω_t^k is the local model from k at round t, n_k is the number of data samples on client k, $n = \sum_k n_k$ is the total number of samples across all clients.

3. System Design and Functionality

As shown in Figure 2, the system consists of three components: cloud, edge, and client. The cloud monitors training, aggregates model parameters from edge devices, and distributes the updated global model. Edge devices perform local training with their own data and upload updates. The client uses the aggregated model for inference on images, videos, or live streams. This setup enables efficient, privacy-preserving distributed learning by combining cloud and edge capabilities.

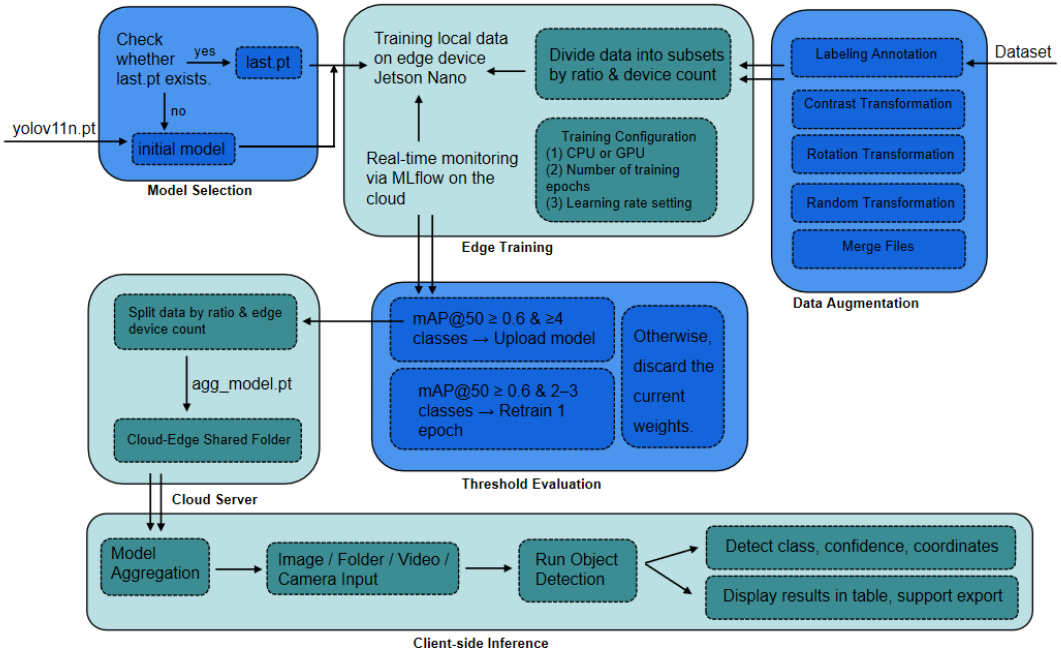

Figure 2: System architecture and workflow

3.1 Cloud

In this system, the cloud functions as the central node of the federated learning architecture, with key responsibilities including monitoring training metrics, aggregating model parameters, and distributing updated models. A local cloud server is used in this study, which connects to the edge devices via a local area network (LAN).

Before training begins, the cloud loads a pre-trained model as the initial global model to ensure that all clients start the first round of federated learning with the same parameters. After completing local training, each edge device uploads its updated model weights to a designated path in a shared directory. As part of the federated learning workflow, the cloud repeatedly checks this directory during each training round. Aggregation only begins once the required number of model files has been uploaded and at least two edge devices have successfully completed training and uploaded their parameters.

After aggregating the models, the cloud saves the new global model as agg_model.pt. In the following round, this updated model replaces the previous one on each edge device. Through continuous iterations, the model is progressively refined with data from different edge nodes, ultimately resulting in a more robust and generalized global model.

Throughout training, the MLflow experiment tracking platform is used to automatically log and visualize key performance metrics—such as loss, mean Average Precision (mAP), precision, and recall—for each round. It also periodically records system resource usage, including CPU and memory utilization, to monitor overall system performance. Figure 3 shows the MLflow monitoring interface.

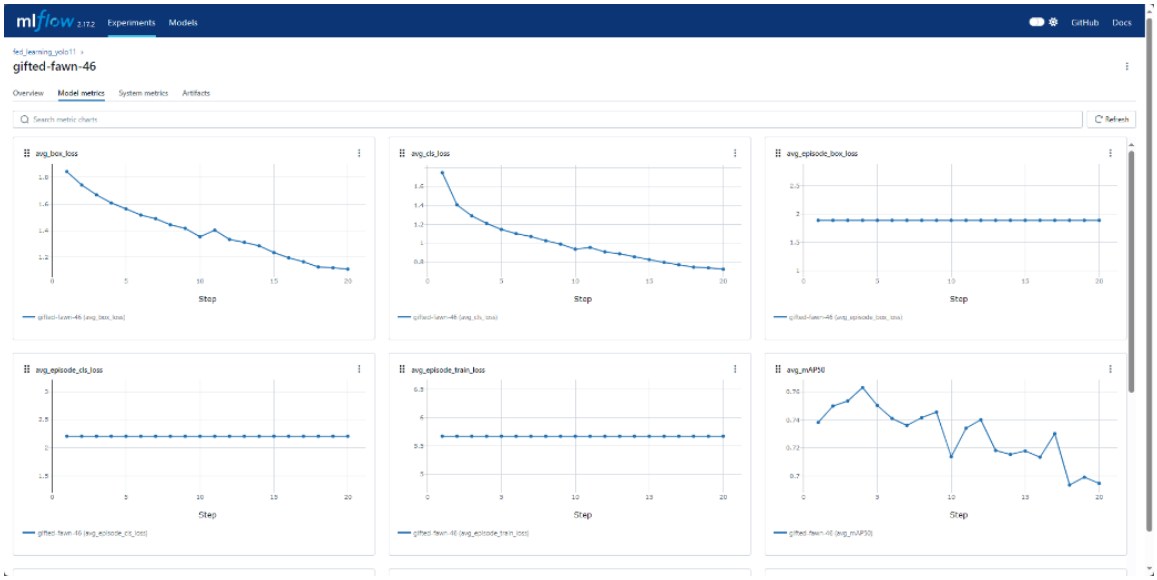

Figure 3: Cloud-Side MLflow Monitoring Interface

3.2 Edge

In this study, three Jetson Nano B01 edge devices were deployed for federated training. Each device is built on the NVIDIA Jetson Nano B01 development board, featuring a quad-core ARM Cortex-A57 CPU and a 128-core CUDA GPU. The devices run Ubuntu 18.04 with PyTorch 1.10, and are connected via HDMI for display and M.2 wireless cards for networking. As key components of the federated learning framework, the edge devices handle receiving model parameters, conducting local training, and uploading updated weights.

Each device trains a YOLOv11 model on its local dataset. Before training, it checks for a local `latest.pt` model file—if found, it is used; otherwise, the initial model `YOLOv11n.pt` is loaded from the shared cloud directory. To ensure training quality, a threshold mechanism is applied. After each round, the model is evaluated based on mAP50 and the number of detected classes. If $mAP50 \geq 0.6$ and the class count ≥ 3, the model is considered qualified and uploaded. If mAP50 falls within [0.2, 0.6] and classes within [2, 3], the training is repeated. If $mAP50 < 0.2$ or classes < 2, the model is discarded and not uploaded.

After uploading, each edge device enters a waiting phase and periodically checks the shared directory for a new aggregated model. Once a new model is detected, it automatically downloads and replaces the previous version, using it as the starting point for the next round of federated training. Each device uploads independently, enabling decentralized yet collaborative model evolution, which enhances overall system intelligence.

The cloud and edge devices operate within the same local area network (LAN), avoiding cross-network communication and ensuring efficient data transfer. Model exchange is managed through a shared directory: the cloud's shared folder is mounted on each Jetson Nano using the NFS protocol. Once mounted, devices can directly access `/cloud_share/` to read the latest aggregated model (`agg_model.pt`) or upload their locally trained models.

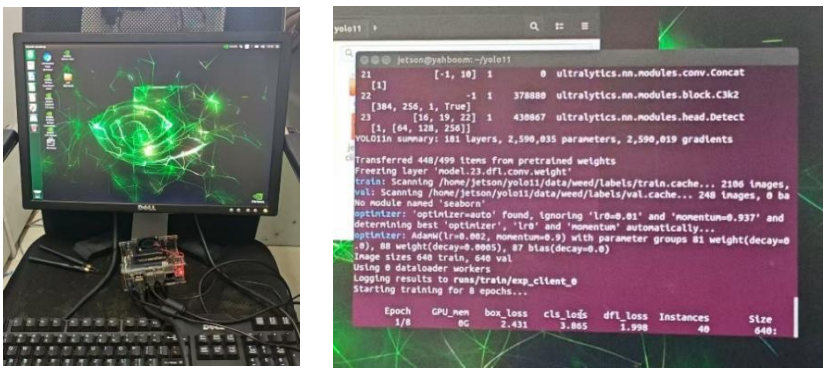

Figure 4: Edge Device Hardware and Training Interface

3.3 Client

In this system, the client is responsible for performing detection and recognition tasks on local images, videos, or real-time camera feeds using the aggregated model produced through federated training on the cloud.
The client is deployed on a desktop computer and features a graphical user interface (GUI) built with PyQt5. As shown in Figure 5, the interface is intuitive and user-friendly, allowing users to view detection results directly. PyQt5 is a Python binding for the Qt framework that supports cross-platform development and provides a wide range of UI components, making it well-suited for rapid desktop application development[7-9].

The client interface includes the following features:
(1) Select Image: Load a local image containing weeds and perform detection using the latest model.
(2) Select Folder: Load a local folder of images and conduct weed detection on each image sequentially.
(3) Open Camera: Activate the local webcam to capture a live frame and perform weed detection.
(4) Image Display Area: Visually displays the detection results in real time.
(5) Export Data: Export the detection results (e.g., class, confidence score, and bounding box coordinates) to an Excel file.

Upon startup, the client checks the shared cloud directory for the latest aggregated model. The client does not participate in model training or parameter uploads. Its purpose is to apply the trained model to real-world scenarios, making the results of edge-side training practically useful and thereby enhancing the overall effectiveness and completeness of the system.

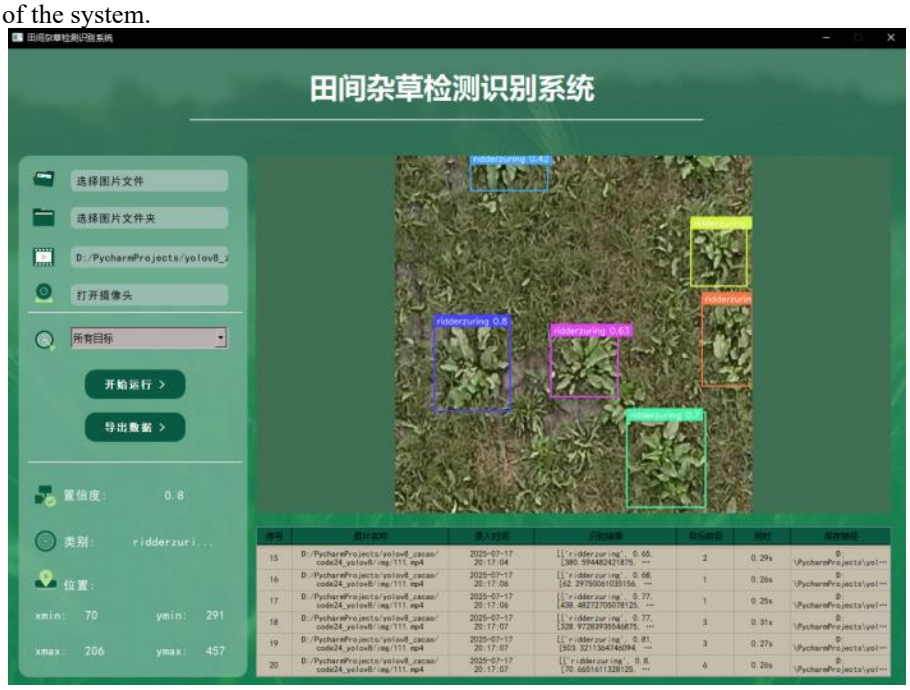

Figure 5: Client Interface

4. Experiments and Result Analysis

4.1 Experimental Setup and Dataset

To evaluate the proposed federated object detection system, three edge devices were configured with a learning rate of 0.001 (decaying to 0.0001), a batch size of 16, and 100 training epochs. A self-collected weed dataset was used, consisting of approximately 1,800 raw images, which were manually filtered and augmented (e.g., flipping, color jitter), resulting in 2,479 images in YOLO format.

The dataset was evenly divided into three subsets, each deployed on a separate device and further split into training, testing, and validation sets. All data remained local to each device to ensure privacy.The subsets included images with varying angles and clarity to simulate real-world conditions. This setup improves system realism and demonstrates its effectiveness in practical, distributed deployment scenarios.

4.2 Experimental Results and Analysis

4.2.1 Evaluation Metrics

All three edge devices utilize the YOLOv11 model to train on their respective datasets. To evaluate the system's performance in object detection and recognition, this study focuses on the following key evaluation metrics:

(1)P Curve: The P curve shows the precision for each class, helping assess how well the model detects individual object categories. It highlights which classes are accurately recognized and which may have more false positives. As shown in Figure 6, the P curve reflects strong precision in weed detection, indicating the model effectively identifies weeds with few incorrect predictions.

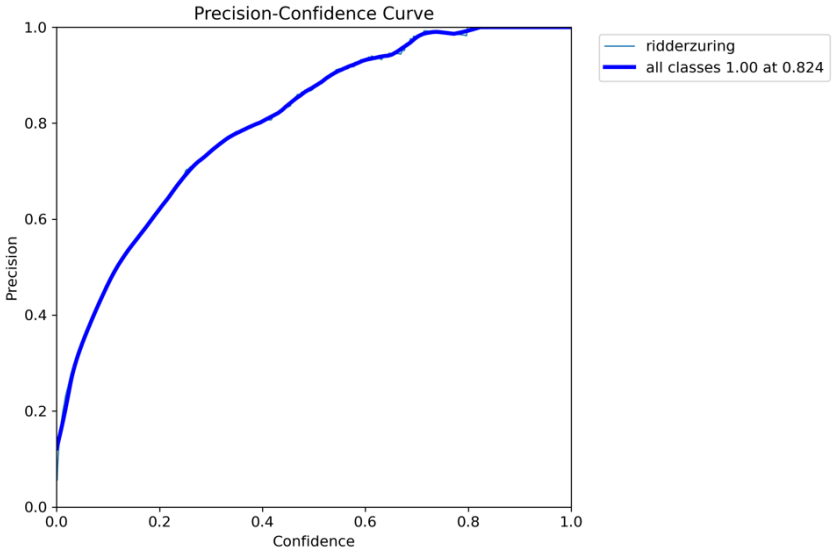

Figure 6: P Curve

(2)R Curve: The R curve shows the recall for each class, indicating the system's ability to identify weed instances comprehensively. As shown in Figure 7, the system performs well in most cases but shows a slight drop in recall for low-resolution or blurred images. Combined with the P curve, the R curve offers a more complete view of the model's detection performance by balancing accuracy and coverage.

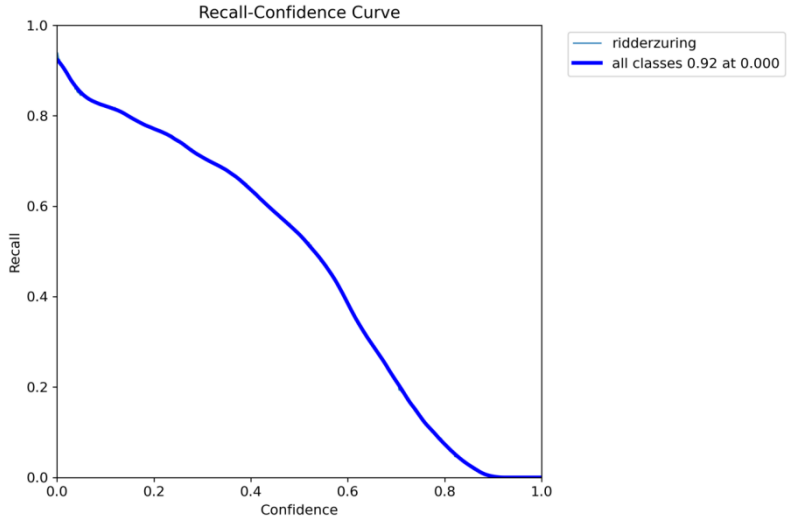

Figure 7: R Curve

(3)PR Curve: The PR curve shows the trade-off between precision and recall at various confidence thresholds. A curve closer to the top-right corner indicates better performance. As shown in Figure 8, the system achieves an average precision (AP) of 0.757 in weed detection, demonstrating balanced and reliable results. The PR curve

offers a comprehensive view of both the accuracy and completeness of the model's predictions.

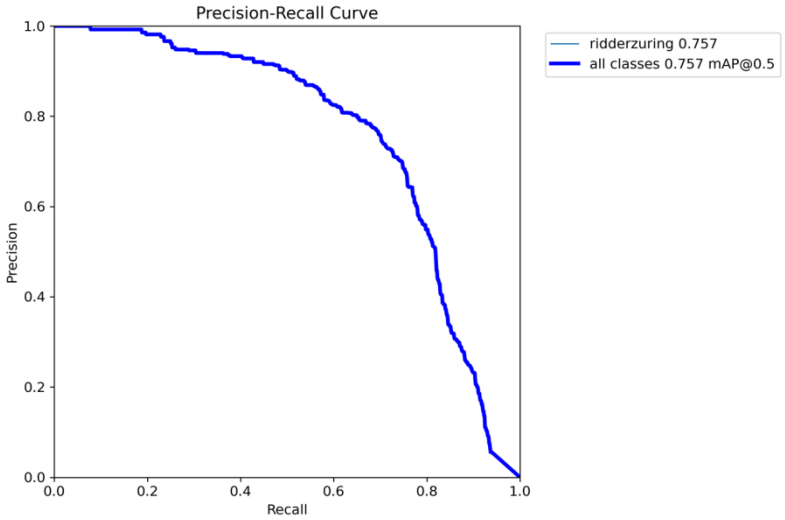

Figure 8: P-R Curve

4.2.2 Single-Device Training Results

To demonstrate the impact of federated learning on object detection performance, this study compared single-device training with federated training using three edge devices. As shown in Figure 9, the single-device results include key metrics such as loss, precision, recall, and mAP, serving as a baseline for assessing the improvements achieved through federated learning in terms of accuracy, robustness, and generalization.

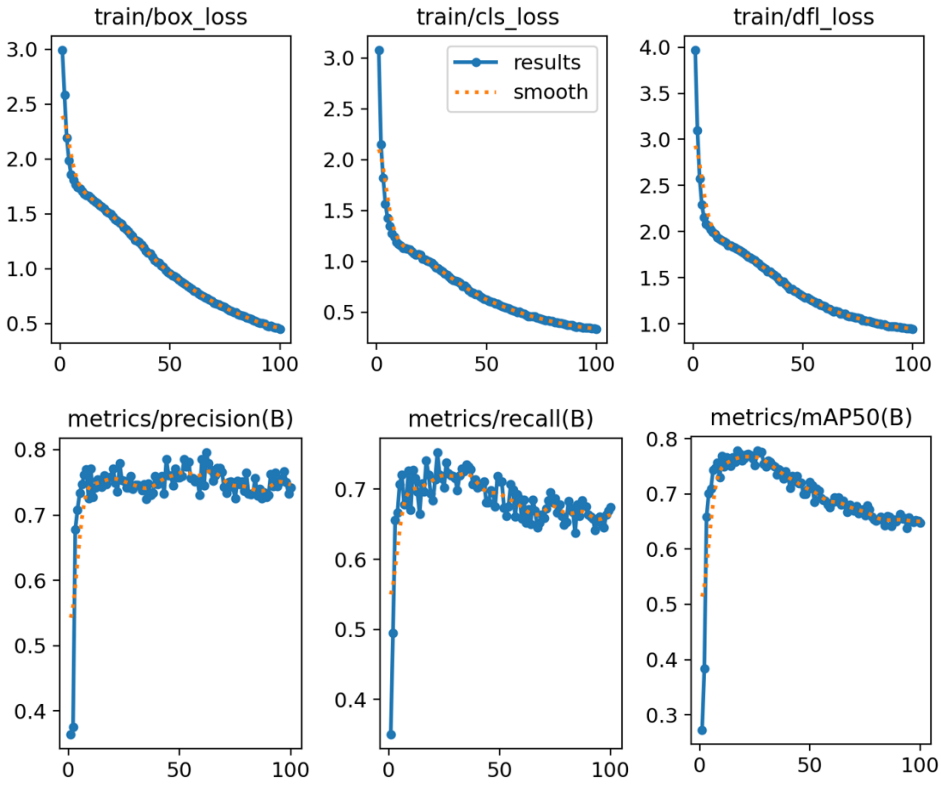

Figure 9: Single-Device Training Results

4.2.3 Federated Learning Training Results

This experiment compared centralized training on a single device with federated training across three edge devices for object detection. The results show that federated training outperformed centralized training in all key metrics:

precision increased from 77.89% to 82.17%, recall from 72.25% to 73.56%, and mAP from 77.82% to 81.45%. These improvements demonstrate that federated training effectively leverages data from multiple devices, enhancing generalization and detection accuracy. The notable gains in precision and mAP highlight its advantage in real-world, collaborative scenarios.

Table 1: Comparative Training Results

	P/%	R/%	mAP/%
Centralized Training on a Single Device	77.89	72.25	77.82
Federated Training with Three Devices	82.17	73.56	81.45

4.3 System Detection Results

To verify the effectiveness of the federated learning system, inference was performed on validation set images. The system detected weeds and output confidence scores, as shown in Figure 10. Detected weeds were marked with red bounding boxes and labeled with confidence values. These results confirm the model's ability to accurately identify and localize targets, demonstrating the practical feasibility of the proposed approach.

Figure 10: Weed Detection Results

5. Conclusion

This study presents a federated learning-based object detection system for weed recognition in smart agriculture, featuring a three-tier cloud-edge-client architecture. By combining the YOLO algorithm with federated learning, the system enables distributed training across multiple edge devices while preserving data privacy and supporting cloud-based model aggregation and updates.

A key innovation is the introduction of a training evaluation and filtering mechanism, using mAP50 and class count thresholds to ensure only high-quality models are aggregated. The system also includes a PyQt5-based GUI on the client side, enabling intuitive inference on local images, videos, and real-time camera feeds, enhancing both usability and practicality.

Despite the progress made, there are still several areas for future improvement:

1. Heterogeneous Device Adaptability: Currently, edge devices are based primarily on Jetson Nano. Future work could extend support to a wider range of embedded platforms and operating systems.

2. Advanced Federated Optimization Algorithms: Techniques such as FedAvgM or FedProx could be introduced to improve model convergence under non-IID (non-independent and identically distributed) data conditions.

3. Model Compression and Acceleration: To accommodate resource-constrained devices, methods like model pruning and quantization could be explored to further enhance inference efficiency.

4. Extension to More Complex Agricultural Scenarios: The system can be extended to tasks such as pest and disease identification or fruit ripeness detection, increasing its generalizability and application value.

In summary, the proposed system not only introduces a novel paradigm for distributed training in agricultural object detection but also provides a practical pathway for deploying federated learning in real-world applications, demonstrating both research significance and practical potential.

References

[1] Redmon J, Farhadi A. YOLOv4: Optimal Speed and Accuracy of Object Detection [EB/OL]. arXiv:2004.10934.
[2] Yang H, Liu M, Wu X, et al. FedVision: An Online Visual Object Detection Platform Powered by Federated Learning [C]//AAAI 2021.
[3] Zhou Y, Wu T, et al. FedDet: A Federated Learning Framework for Object Detection [C]//NeurIPS FL Workshop, 2021.
[4] Li T, Sahu A K, Talwalkar A, Smith V. Federated Learning: Challenges, Methods, and Future Directions[J]. IEEE Signal Processing Magazine, 2020.
[5] Kairouz P, et al. Advances and Open Problems in Federated Learning[J]. Foundations and Trends® in Machine Learning, 2021.
[6] Chen M, Yang J, et al. A Survey on Federated Learning in Edge Computing: Research Problems and Solutions[J]. IEEE Network, 2021.
[7] Aidil Saputra Kirsan, Kosuke Takano, Sallie Trixie Zebada Mansurina.EksPy: A new Python framework for developing graphical user interface based PyQt5[J]. International Journal of Electrical and Computer Engineering (IJECE), 2024, 14(1): 520–531. DOI: 10.11591/ijece.v14i1.pp520-531.
[8] Zheng, Y., Zhou, G., & Lu, B. Rebar Cross-section Detection Based on Improved YOLOv5s Algorithm[J]. Innovation & Technology Advances, 2023, 1(1): 1–6. https://doi.org/10.61187/ita.v1i1.1
[9] Zhao, X., Zhang, L., & Hu, Z. Smart warehouse track identification based on Res2Net-YOLACT+HSV[J]. Innovation & Technology Advances, 2023, 1(1): 7–11. https://doi.org/10.61187/ita.v1i1.2