SCIENTIFIC OASIS

Decision Making: Applications in Management and Engineering

Journal homepage: www.dmame-journal.org
ISSN: 2560-6018, eISSN: 2620-0104



Adaptive Edge Cloud Decision Support for Apple Disease Detection Using Deep Learning

Akniyet Nurzhaubayev[1], Seilov Shakhmaran[1], Bibinur Zhursinbek[1], Yerden Zhursinbek[1], Stefan Gubo[2], Amir Mosavi[2,3,4,5,*]

1 Institute of Digital Sciences and Artificial Intelligence, Eurasian National University, Astana, Kazakhstan.
2 Faculty of Economics and Informatics, Selye Janos University, Komarno, Slovakia.
3 John von Neumann Faculty of Informatics, Obuda University, Budapest, Hungary.
4 Astana IT University, Astana, Kazakhstan.
5 Ludovika University of Public Service, Budapest, Hungary.

ARTICLE INFO

Article history:
Received 15 November 2025
Received in revised form 19 April 2026
Accepted 17 May 2026
Available online 30 June 2026

Keywords:
Adaptive Decision-Making, Edge-Cloud Decision Support, Orchard Disease Detection, Decision-Making, IoT, Edge Computing, Soft Computing, YOLOv8, Deep Learning, Object Detection, Artificial Intelligence

ABSTRACT

The timely identification of diseases is a fundamental requirement for effective orchard management. This study introduces an adaptive decision support system for automated apple disease identification that integrates deep learning with a hybrid edge-cloud computing architecture. The proposed framework dynamically determines the location and execution strategy of inference based on prediction confidence. To facilitate model development and assessment, a domain-specific dataset comprising leaf and fruit images was established, with annotations provided by professional biologists. A quantised YOLOv8n model was implemented on a cost-effective edge device equipped with a neural processing unit (NPU), whereas a larger YOLOv8l model was deployed in the cloud to undertake further analysis. The framework incorporates a confidence-driven decision mechanism that retains predictions exhibiting high confidence at the edge, forwards ambiguous cases to the cloud for further assessment, and discards observations whose confidence levels fall below the acceptable threshold. Through this mechanism, prediction confidence is translated into actionable decisions governing computational resource allocation. Experimental validation using an independent orchard test dataset demonstrates the reliability of the proposed disease detection framework and its superior performance relative to conventional approaches based exclusively on edge or cloud inference. The adaptive architecture enhances detection accuracy whilst simultaneously decreasing bandwidth requirements and end-to-end response latency. In particular, the majority of images are analysed locally, with cloud-based processing reserved for cases characterised by uncertain predictions. These findings establish that confidence-guided adaptive decision-making can concurrently enhance detection accuracy, communication efficiency and response time. Accordingly, the proposed Edge-Cloud Decision Support approach offers an intelligent and computationally efficient framework for precision agriculture and Internet of Things (IoT) applications.

* *Corresponding author.*
E-mail address: amir.mosavi@nik.uni-obuda.hu

https://zenodo.org/records/24353758

1. INTRODUCTION

Fruit production in orchard systems represents an important component of global agricultural activity, as crop health directly influences yield stability, food security and economic sustainability. Accordingly, orchard management involves a complex decision-making process characterised by multiple interconnected considerations [1]. Adaptive decision-making enables management strategies to respond promptly to variations in disease occurrence and computational demands. Within this context, intelligent decision systems and edge-cloud-based decision-support architectures offer reliable and efficient solutions for addressing the challenges associated with orchard disease management [2].

The occurrence of disease outbreaks in apple orchards can lead to substantial agricultural and economic losses, making early identification particularly important for enabling timely intervention and preventive disease control. Accurate and timely detection can also help limit the excessive application of chemicals and reduce the operational costs associated with orchard management [3]. Furthermore, computer vision technologies facilitate the continuous observation and assessment of crop conditions in natural field environments. Their application consequently strengthens precision-oriented orchard management and contributes to more sustainable agricultural production practices [4]. Advances in deep learning and computer vision have significantly enhanced the capabilities of automated plant disease identification systems. In particular, YOLOv8 combines rapid inference with precise disease localisation, whereas lightweight convolutional neural network (CNN) architectures have demonstrated competitive detection performance when evaluated using curated datasets [3; 5]. However, datasets developed under controlled experimental conditions do not necessarily reflect the diverse environmental characteristics encountered in operational orchards. Variations in lighting conditions, complex visual backgrounds and differences in disease appearance may therefore compromise the reliability of automated detection. Edge-cloud computing provides an alternative computational architecture through the integration of local processing capabilities and remote cloud resources. Promising results have been documented for edge-based agricultural detection applications [6]. Nevertheless, the implementation of these technologies in operational orchard environments continues to present considerable challenges owing to practical field constraints [7].

This study primarily contributes an adaptive decision-making framework specifically designed for orchard disease detection. Through the proposed Edge-Cloud Decision Support mechanism, the framework dynamically identifies the most suitable computational environment for executing inference based on the confidence of individual predictions. Predictions that achieve an adequate confidence level are processed locally at the edge, whereas cases requiring further evaluation are directed to cloud-based inference. Consequently, the proposed framework broadens the scope of conventional disease detection by incorporating computational decision-making alongside detection accuracy. Rather than relying on a fixed allocation of computational resources, the system dynamically determines the appropriate processing strategy according to prediction uncertainty and prevailing operational conditions. This integration establishes an adaptive decision-making approach intended to facilitate the practical implementation of intelligent agricultural decision-support systems.

The deployment of lightweight deep learning architectures often entails a decline in detection accuracy following quantisation and model compression. Consequently, implementing such models on edge devices may compromise the reliability of Orchard Disease Detection. Although cloud-based inference can offer improved detection performance through access to more powerful computational resources, its operation depends on network availability. Moreover, communication delays associated with data transmission may adversely affect real-time Orchard Disease Detection. These constraints are particularly pronounced in geographically isolated agricultural regions, where

network connectivity is frequently unreliable. In addition to computational and communication limitations, environmental variations introduce further challenges to detection reliability. Changes in illumination, overlapping leaves and interference from complex backgrounds increase visual uncertainty, while the identification of subtle symptoms during the early stages of infection makes Orchard Disease Detection considerably more challenging [8]. Consequently, achieving consistently reliable disease identification under practical field conditions remains difficult.

An important research gap concerns the development of Adaptive Decision-Making strategies for hybrid inference architectures. Most existing investigations concentrate primarily on improving model accuracy or enhancing the computational efficiency of edge-based implementations, whereas comparatively limited consideration has been given to Edge-Cloud Decision Support mechanisms. Such mechanisms are required to dynamically coordinate inference execution according to prediction uncertainty while accounting for the computational limitations of individual devices. Furthermore, the adoption of domain-specific datasets annotated by experts remains inconsistent across existing investigations [9]. This shortcoming may restrict the generalisation capability of trained models across different orchard environments and climatic conditions. Accordingly, effective Adaptive Decision-Making requires not only reliable disease identification but also validation within the relevant environmental and operational context. The development of scalable hybrid architectures capable of explicitly incorporating prediction uncertainty therefore remains an insufficiently investigated research direction.

Recent investigations into edge computing demonstrate that deploying disease detection models on platforms such as Raspberry Pi and Jetson Nano can facilitate real-time agricultural monitoring at relatively low cost while maintaining acceptable detection accuracy [5]. Similarly, hybrid edge-cloud architectures, which perform preliminary analysis locally and delegate computationally demanding assessments to remote cloud infrastructure, have demonstrated improvements in processing efficiency and system scalability. Further studies have identified reductions in latency, more efficient bandwidth utilisation and improved operational resilience under unstable network connectivity in agricultural environments [6]. Collectively, these findings establish the technical feasibility and practical relevance of integrating edge and cloud resources within agricultural disease detection pipelines. A further unresolved issue involves the development of intelligent offloading mechanisms capable of determining whether locally generated predictions should be accepted, transferred to cloud-based inference or rejected. Existing approaches frequently rely on predefined static confidence thresholds, which restrict their adaptability and may increase false-positive detections under changing field conditions. In addition, optimisation methods, including pruning, quantisation and knowledge distillation, are commonly investigated as separate techniques rather than being incorporated into jointly coordinated hybrid inference architectures [10]. Consequently, the simultaneous optimisation of detection accuracy, processing latency and bandwidth consumption has not received sufficient attention, despite its importance for reliable deployment in operational orchard environments.

Empirical findings further suggest that combining advanced model optimisation with hybrid inference architectures can improve the performance of agricultural monitoring systems. Fine-tuned cloud-edge detection frameworks have demonstrated improvements in detection accuracy and operational efficiency across smart farming applications. Furthermore, systematic reviews highlight the importance of adaptive, multi-stage detection strategies that explicitly account for environmental variability and prediction uncertainty. Integration with Internet of Things (IoT) infrastructures additionally facilitates distributed data collection and supports scalable implementation across multiple orchard locations. Collectively, these developments provide a rationale for designing computational architectures that integrate lightweight embedded detection

with high-capacity verification mechanisms.

Within the specific operational setting considered in this study, observations conducted in the Issyk Aport apple orchard are affected by complex visual backgrounds, seasonal environmental variations and intermittent network connectivity. These characteristics necessitate the development of domain-specific datasets and robust deployment strategies capable of accommodating local field conditions [11]. Accurate labelling by agricultural specialists is particularly important, given that disease symptoms may be subtle and difficult to distinguish during their initial development. Accordingly, the study objectives focus on designing a real-time disease detection approach that maintains high detection accuracy without compromising low-latency processing under practical orchard conditions. The principal aim is to develop a hybrid edge-cloud computing architecture in which a quantised YOLOv8n model operates on embedded hardware and selectively transfers uncertain predictions to a high-capacity cloud-based detector according to prediction confidence. This configuration is intended to facilitate accurate, bandwidth-efficient and real-time apple disease detection. The timely identification of apple diseases is fundamental to minimising crop losses and strengthening orchard management practices. Edge computing and hybrid edge-cloud architectures are increasingly being investigated as viable approaches for achieving real-time agricultural disease detection. In particular, integrating the Internet of Things (IoT) with artificial intelligence (AI) at the edge enables remote crop monitoring, reduces processing latency and facilitates immediate decision-making [1]. The application of mobile edge AI for disease identification and mapping in citrus orchards further illustrates the advantages of performing inference locally using low-power computational devices. Similarly, lightweight neural network architectures implemented on Arduino-based edge platforms have demonstrated the feasibility of efficient, real-time apple disease classification at the orchard level. Reviews examining lightweight deep learning architectures also emphasise the importance of computationally efficient models for enabling real-time disease identification in resource-constrained environments [3]. A broad range of deep learning architectures, including You Only Look Once (YOLO), Faster Region-based Convolutional Neural Network (Faster R-CNN), MobileNet, EfficientNet and NASNet-Mobile, have been employed for fruit disease detection. Their detection performance varies according to the selected dataset and the preprocessing procedures applied. Architectures such as YOLOv4, YOLOv8, Inception-ResNet-V2, VGG-INCEP and AlexNet have demonstrated different levels of detection accuracy across agricultural imaging applications. Among these approaches, the lightweight YOLOv8n architecture offers low-latency inference on edge devices while retaining high detection accuracy, making it a suitable candidate for deployment within hybrid edge-cloud computing frameworks [5].

The comparative findings summarised in Table 1 indicate that previous investigations have predominantly focused on maximising cloud-based detection accuracy or improving computational efficiency at the edge, thereby revealing a methodological limitation in existing approaches. Although hybrid architectures integrate both computational resources, the mechanisms governing inference allocation remain relatively underdeveloped. In particular, prediction uncertainty is seldom translated into explicit operational decisions, such as accepting predictions locally, escalating uncertain cases to cloud-based processing or rejecting unreliable observations. Consequently, the trade-offs associated with computational resource utilisation are not adequately regulated. The proposed framework addresses this limitation by introducing confidence-driven decision support, in which prediction confidence serves as the principal decision variable. Predictions associated with sufficiently high confidence are retained and accepted locally, whereas uncertain observations are forwarded to the cloud for additional assessment and predictions exhibiting inadequate confidence are rejected. This decision policy facilitates adaptive computational resource allocation according to prediction reliability. Therefore, the principal novelty of the proposed approach resides in its

uncertainty-aware decision-support mechanism rather than merely in the integration of hybrid edge-cloud computing. Furthermore, Table 1 highlights the combination of several important capabilities within the proposed framework, including domain-specific data utilisation, low-cost edge-based inference, real-time processing, bandwidth-conscious operation and adaptive inference routing.

Table 1
Comparison of Recent State-of-the-Art Plant Disease Detection Approaches with this Work

Work	Deployment	Custom Dataset	Low Computational Cost	Real-time	Latency	Bandwidth	Accuracy
Liu et al. [7]	Cloud	✗	✗	✗	✗	✗	✓
Wang et al. [8]	Edge	✗	✗	✓	✓	✗	✓
Zou et al. [9]	Edge	✗	✓	✓	✓	✗	✓
Liu et al. [10]	Cloud	✓	✗	✗	✗	✗	✓
Karim et al. [11]	Edge	✗	✗	✓	✓	✓	✓
Proposed (this work)	Hybrid	✓	✓	✓	✓	✓	✓

Ensemble learning provides an effective means of enhancing predictive accuracy and model robustness through the integration of multiple classification algorithms, including support vector machines (SVM), k-nearest neighbours (KNN) and random forest classifiers (RFC). An apple disease identification framework based on ensemble learning has demonstrated satisfactory classification accuracy, highlighting the potential of this approach for addressing the complexity of orchard environments [12]. Similarly, adaptive multi-stage ensemble learning has improved precision in tomato leaf disease classification, indicating its potential applicability to apple disease identification [13]. Beyond classifier selection, image preprocessing and feature extraction are fundamental to achieving reliable detection performance. Methods such as Gaussian smoothing, adaptive histogram equalisation, K-means clustering and Watershed segmentation contribute to image enhancement and the isolation of disease-related visual characteristics [14]. Moreover, integrating complementary features within deep hierarchical architectures can further improve detection performance, with reported accuracy reaching 99.6 across multiple fruit disease categories [15].

Hybrid edge-cloud computing architectures offer a means of overcoming limitations associated with edge devices, particularly restricted processing capabilities and variations in environmental conditions. The integration of contextual data fusion with faster-CNN architectures and selective cloud offloading has demonstrated potential for accurate disease identification and real-time orchard monitoring [16; 17]. Furthermore, wireless sensor networks combined with edge computing facilitate continuous disease surveillance and data transmission, enabling subsequent high-accuracy analysis [18]. Despite these developments, achieving reliable real-time disease detection remains challenging because of limitations in dataset availability and the considerable variability of agricultural environments. Access to diverse, high-quality datasets is therefore essential for strengthening model generalisation and improving detection performance [19]. Incorporating IoT-based sensing technologies and hyperspectral imaging may provide additional information for improving disease identification accuracy and advancing the development of intelligent orchard systems [13; 19].

Overall, the existing literature indicates that combining hybrid edge-cloud architectures with lightweight detection models, such as YOLOv8n, ensemble learning techniques and advanced image preprocessing methods offers a promising computational framework for real-time apple disease identification. The integration of low-latency inference at the edge with selective cloud-based processing can enhance detection accuracy, improve system responsiveness and facilitate more efficient utilisation of computational resources, thereby supporting intelligent and sustainable orchard management.

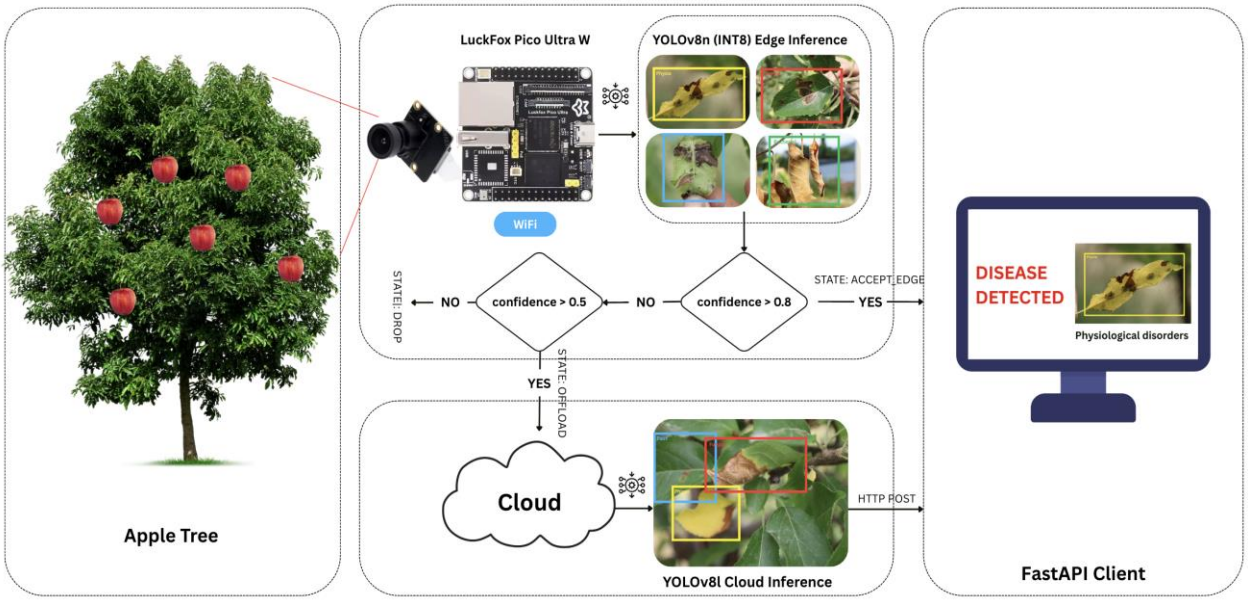


Fig.1: Proposed Architecture

1.1 Literature Analysis

A considerable body of research has investigated lightweight convolutional neural network (CNN) architectures to facilitate real-time plant disease identification on computationally constrained devices. Although these investigations have made valuable contributions to model optimisation, detection accuracy and embedded deployment, their methodologies differ considerably in terms of computational implementation, experimental validation and practical applicability.

Liu et al. [7] developed a two-stage segmentation framework, designated L-DPNet+D-UNet, for detecting frog-eye leaf spot by independently addressing leaf segmentation and lesion segmentation. Their proposed methodology achieved a mean Intersection-over-Union (mIoU) of 91.27%, mean pixel accuracy (mPA) of 94.32% and severity classification accuracy of 94.81%. The experimental dataset comprised 1,372 individual leaf images obtained from the Plant Pathology 2020/2021 competition. These images underwent manual pixel-level annotation, followed by data augmentation using the Mosaic technique, resulting in an expanded dataset of 2,192 images. The study demonstrates substantial annotation effort and rigorous pixel-level performance evaluation. Nevertheless, its implementation remains restricted to offline processing, without experimental deployment on embedded or edge computing platforms. Furthermore, neither inference latency nor bandwidth requirements were investigated. The reliance on isolated leaf images also represents a methodological constraint, as such images present considerably fewer visual complexities than continuous orchard video streams, where leaf occlusion and changing illumination can affect segmentation performance.

Wang et al. [8] introduced ELM-YOLOv8n, an enhanced lightweight object detection architecture incorporating FasterNet blocks, exponential moving average (EMA), depthwise separable convolutions with dynamic overhead (DESCS-DH) and normalised Wasserstein distance (NWD). The proposed model achieved a mean average precision at an intersection-over-union threshold of 0.5 (mAP@0.5) of 96.7% and an F1-score of 94.0%. Compared with the baseline YOLOv8n architecture, the model reduced the number of parameters by 44.8% and required 39.5 GFLOPs. Ablation experiments additionally established the contributions of EMA and NWD, with their integration producing an F1-score improvement of approximately 1.2 percentage points. These architectural modifications represent a valuable engineering contribution, particularly for detecting small disease-

related targets in orchard environments. However, the investigation remains primarily simulation-oriented. Although the authors identified UAVs and mobile devices equipped with GPUs containing at least 2 GB of video memory as potential deployment platforms for future work, no empirical evaluation on embedded hardware was presented. Similarly, inference latency and bandwidth consumption were not quantified. Furthermore, the dataset was predominantly assembled from verified public image repositories rather than continuous image acquisition under operational orchard conditions.

Zou et al. [9] proposed LightYOLO-AppleLeafDx, a YOLOv8-derived detection architecture specifically optimised for deployment on embedded computing platforms. A particularly important contribution of this investigation is its practical implementation on the Rockchip RV1103 (LuckFox Pico Plus) using INT8 quantisation. The optimised model attained an mAP@0.5 of 0.965 and an mAP@0.5:0.95 of 0.587, while its compressed model size was reduced to 5.2 MB. Regarding computational performance, inference speeds of 107.2 frames per second (fps) on an RTX 2080 Ti and 14.8 fps on the RV1103 neural processing unit (NPU) were reported, demonstrating its suitability for near-real-time embedded inference. The authors also documented the complete model conversion workflow, encompassing PyTorch, Open Neural Network Exchange (ONNX) and Rockchip Neural Network (RKNN) formats. Despite these achievements, certain edge experiments relied on instrumentation-based emulation, and extensive validation under actual orchard field conditions was not provided. Moreover, the evaluation dataset was derived from SCIDB and Kaggle repositories rather than continuous image streams representative of operational orchard environments. The influence of network bandwidth limitations was also excluded from the investigation. Nevertheless, this study constitutes a particularly relevant precedent for the present research because of its implementation of a YOLOv8-based architecture on Rockchip hardware.

Liu et al. [10] investigated apple mosaic virus (ApMV) severity assessment through the integration of hyperspectral imaging and chlorophyll content inversion. Their experimental investigation involved 360 leaf samples collected from orchards and employed an SOC-710VP hyperspectral camera operating across a spectral range of 374.8 to 1042.1 nm with 128 channels, alongside chlorophyll measurements obtained using a Dualex-4 instrument. Using a stacking-based boosting model, the researchers achieved a validation coefficient of determination (R^2) of 0.9644 and an overall classification accuracy of 98.89%. This investigation provides a scientifically robust approach to disease severity assessment through physiologically relevant measurements. However, hyperspectral imaging involves comparatively expensive equipment and is not readily suited to real-time implementation on resource-constrained embedded platforms. Consequently, the methodology presents limited applicability to low-cost orchard monitoring systems. In addition, latency and bandwidth performance were outside the scope of this investigation.

Karim et al. [11] developed a real-time grape leaf disease classification framework implemented on an NVIDIA Jetson Nano using a lightweight CNN architecture combined with the Gradient-weighted Class Activation Mapping (Grad-CAM) interpretability technique. The investigation demonstrated several practical advantages, including actual edge-device deployment, a real-time graphical user interface and systematic power consumption profiling. During inference, the reported average power consumption was approximately 3.4 W. The authors also evaluated inference time using TensorFlow Lite (TFLite) models and examined the relationship between computational complexity, expressed in GFLOPs, and processing latency. Such detailed reporting of hardware-level performance and energy consumption remains relatively uncommon in agricultural artificial intelligence research. Nevertheless, the framework is restricted to disease classification rather than object detection, utilises the Kaggle Grapevine dataset and performs all computational operations locally. Consequently, neither hybrid inference mechanisms nor cloud-based task offloading and

throughput optimisation were considered. Despite the progress demonstrated by these investigations, an important research gap remains concerning the development of hybrid edge-cloud architectures that incorporate selective task offloading according to prediction confidence, reliability or task difficulty. In particular, previous studies have not systematically compared the latency distributions of local device-based inference with the round-trip communication times associated with cloud offloading. Dataset limitations represent another recurring concern, as most investigations have relied on curated or laboratory-oriented image collections, with relatively little emphasis on continuous orchard video acquisition under natural illumination and occlusion conditions. Furthermore, comprehensive reporting of latency and throughput remains uncommon. Among the reviewed investigations, Karim et al. [11] specifically reported energy consumption of approximately 3.4 Won the Jetson Nano, whereas Zou et al. [9] provided on-device inference speed measurements for Rockchip NPU hardware. The research objectives addressed in the existing literature are also distributed across different computational tasks. Specifically, segmentation-based approaches were investigated by Liu et al. [7] and Liu et al. [10], object detection was addressed by Wang et al. [8] and Zou et al. [9], whereas Karim et al. [11] concentrated on disease classification. Collectively, these studies demonstrate substantial advances in lightweight model development, segmentation performance and the implementation of deep learning architectures on embedded platforms. Nevertheless, the limited integration of hybrid edge-cloud computing represents a significant unresolved challenge for practical Internet of Things (IoT) applications. Edge-based systems may encounter restrictions in detection capability and computational capacity, whereas cloud-dependent approaches remain vulnerable to communication latency and throughput limitations. The absence of coordinated inference strategies consequently restricts the effectiveness of these approaches under practical agricultural operating conditions.

The present study addresses these limitations through the development of a YOLOv8-based hybrid edge-cloud infrastructure for apple disease detection, implemented on LuckFox Pico Ultra W and evaluated using an orchard-specific dataset. Table 1 presents a comparative assessment of the principal characteristics and capabilities of the reviewed investigations in relation to the proposed framework. Previous research establishes that lightweight deep learning architectures combined with edge computing can facilitate real-time apple disease identification. However, significant limitations persist concerning practical deployment and comprehensive system-level evaluation. Most existing approaches prioritise either detection accuracy through cloud-based processing or computational efficiency through local edge inference, while coordinated edge-cloud operation and confidence-guided task offloading remain insufficiently investigated. Moreover, latency and bandwidth consumption are rarely evaluated under authentic orchard conditions. The widespread use of curated and static datasets also limits the extent to which existing findings represent continuous agricultural environments characterised by natural occlusion and illumination variability.

To address these methodological and deployment-related limitations, this study introduces a hybrid edge-cloud disease detection framework based on YOLOv8. The proposed architecture is implemented on a resource-constrained embedded platform and evaluated using an orchard-specific dataset acquired under realistic environmental conditions. Its principal contribution involves integrating lightweight disease detection, adaptive computational offloading and comprehensive system-level performance assessment. The evaluation encompasses detection accuracy, inference latency and throughput, thereby providing a broader assessment of the framework's suitability for practical real-time apple disease monitoring in intelligent orchard environments. Importantly, the methodological novelty of the proposed framework is associated primarily with its adaptive decision-support capability rather than the integration of hybrid computing resources alone. Prediction confidence functions as the principal decision variable governing both the execution location and the

allocation of computational resources. Accordingly, predictions exhibiting high confidence are processed and accepted locally at the edge, whereas uncertain predictions are transferred to the cloud for further verification. Observations associated with very low confidence are rejected rather than being accepted as reliable detections. Through this mechanism, prediction uncertainty is systematically translated into operational decisions governing inference execution and resource utilisation. Thus, the proposed contribution extends beyond conventional disease detection by establishing an intelligent decision-support mechanism for distributed artificial intelligence. The coordinated allocation of computational resources according to prediction reliability provides a unified strategy for improving detection accuracy, reducing latency and bandwidth consumption, and enhancing overall computational efficiency.

2. MATERIALS AND METHODS

2.1 Architecture

The edge computing platform comprises a LuckFox Pico Ultra W, equipped with a 1 TOPS neural processing unit (NPU), an ARM Cortex-A7 central processing unit (CPU) and integrated Wi-Fi connectivity. The device is interfaced with an external 3 MP SC3336 MIPI-CSI RGB camera, which continuously acquires images of apple foliage. Disease detection is performed using a YOLOv8n architecture containing 3.2 M parameters, which is quantised to INT8 through quantisation-aware training (QAT). Before inference, the acquired images are resized to 640×640 pixels, after which the model generates bounding boxes, disease class predictions and corresponding confidence scores. Image preprocessing operations, including resizing and normalisation, together with postprocessing through Non-Maximum Suppression (NMS), are performed by the CPU, whereas the computationally intensive inference stage is executed on the NPU. Furthermore, a lightweight HTTP client is implemented on the edge device to establish direct communication with the cloud application programming interface (API).

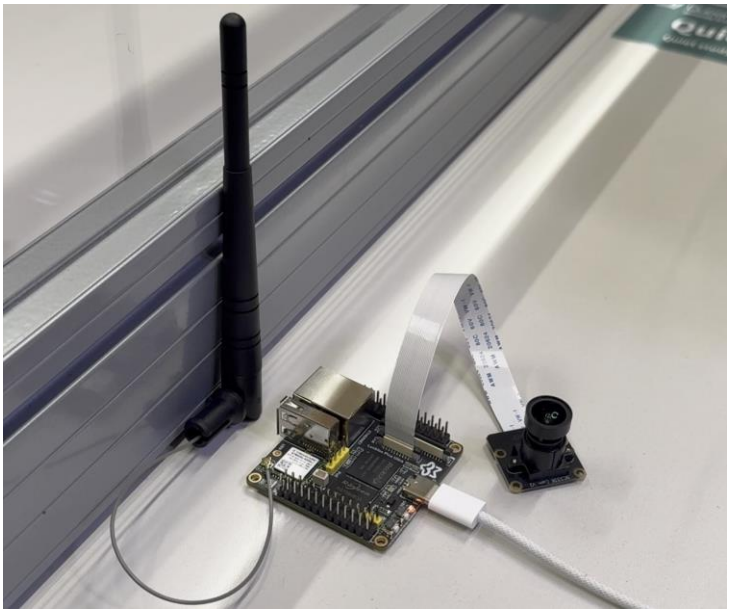


Fig.2: LuckFox Pico Ultra W with CSI Camera

The cloud infrastructure hosts a high-capacity YOLOv8l model, which provides enhanced detection accuracy, particularly when processing complex cases or observations associated with low prediction confidence. The model is deployed on GPU-based computing infrastructure, allowing the

use of larger input resolutions of up to 1024×1024 pixels and deeper processing layers to facilitate fine-grained disease detection. A FastAPI-based server receives image data transmitted from the edge device through HTTP, executes the detection process and returns the refined prediction results. Communication between the edge and cloud components is established through HTTP over the integrated Wi-Fi module of the LuckFox Pico Ultra W. The system implements a dual-threshold decision policy to determine the appropriate inference pathway. When the confidence score generated by the edge model exceeds 80%, the detection results are forwarded directly to the farmer's dashboard without involving cloud processing. Conversely, when the confidence score falls within the range of 50% to 80%, the corresponding image is transmitted to the FastAPI endpoint through an HTTP POST request for further cloud-based analysis.

2.2 Algorithm

Let $F_t \in \mathbb{R}^{H\times W\times 3}$denote the RGB frame captured at time t. The edge detection model, E, implemented using the RKNN-optimised YOLOv8n architecture, processes the input frame and generates a set of detections defined as follows:

$$E(F_t) = \{(b_i, c_i, p_i)\}_{i=1}^{N_t} \tag{1}$$

where b_i represents the bounding box coordinates, $c_i$denotes the corresponding class label, and $p_i \in [0, 1]$indicates the associated prediction confidence. The overall confidence of each frame is subsequently determined by selecting the highest confidence value among all detected objects:

$$\hat{p}_t = \max_{1\le i\le N_t} p_i \text{ (with } \hat{p}_t = 0 \text{ if } N_t = 0) \tag{2}$$

To regulate the allocation of inference tasks between the edge and cloud environments, two fixed confidence thresholds satisfying $0 < \tau_1 < \tau_2 < 1$are established as follows:

$$\tau_1 = 0.50,\ \tau_2 = 0.80 \tag{3}$$

The cloud-based detection model, C, implemented using YOLOv8l, performs further analysis when an observation is transferred from the edge and produces the following set of detections:

$$C(F_t) = \left\{\left(\tilde{b}_j, \tilde{c}_j, \tilde{p}_j\right)\right\}_{j=1}^{M_t} \tag{4}$$

For each input frame F_t, the system determines the appropriate inference-routing strategy through a decision variable d_t, which can assume one of three possible states: DROP, OFFLOAD or ACCEPT_EDGE. The routing policy is formally expressed as:

$$d_t = \begin{cases} \text{DROP}, & \hat{p}_t < \tau_1, \\ \text{OFFLOAD}, & \tau_1 \le \hat{p}_t < \tau_2, \\ \text{ACCEPT_EDGE}, & \hat{p}_t \ge \tau_2. \end{cases} \tag{5}$$

The operational behaviour associated with each routing decision is defined as follows:

- DROP: The input frame is discarded without initiating any network communication.
- OFFLOAD: The edge device transmits the corresponding JPEG image to the cloud endpoint /infer through an HTTP POST request. The cloud executes the detection model C, after which the resulting JSON response is forwarded to the farmer's dashboard through the /ingest/cloud endpoint.
- ACCEPT_EDGE: The detection results generated locally by the edge model $E$are immediately transmitted to the farmer's dashboard as a JSON payload through the /ingest/edge endpoint, without requiring cloud-based inference.

Let L_E denote the inference time at the edge device, $L_{\text{net}\to C}$ represent the uplink transmission time from the edge device to the cloud, $L_C$indicate the cloud inference time, and $L_{\text{net}\to D}$ denote the transmission time required to deliver the detection results to the dashboard. Accordingly, the end-to-end latency associated with displaying a detection result depends on the selected routing decision and is expressed mathematically as:

$$L_t = \begin{cases} L_E + L_{\text{net}\to D}, & d_t = \text{ACCEPT_EDGE}, \\ L_E + L_{\text{net}\to C} + L_C + L_{\text{net}\to D}, & d_t = \text{OFFLOAD}. \end{cases} \quad (6)$$

For frames assigned to the DROP category, no detection result is generated or delivered to the dashboard.

Algorithm 1 Edge-Side Hybrid Inference

Require: Frame F, edge model $\mathcal{E}(\cdot)$, thresholds $\tau_1 = 0.50$, $\tau_2 = 0.80$
Ensure: Decision d_t and output detections
D ← $\mathcal{E}(F)$
$\hat{p}$ ← max confidence in D (default 0 if D = ∅)
if $\hat{p} < \tau_1$ then
d_t ← DROP
else if $\tau_1 \leq \hat{p} < \tau_2$ then
Send F via HTTP POST to Cloud endpoint /infer
$D_{\text{cloud}} \leftarrow \mathcal{C}(F)$
d_t ← OFFLOAD
Forward D_{cloud} to Dashboard /ingest/cloud
else if $\hat{p} \geq \tau_2$ then
d_t ← ACCEPT_EDGE
Forward D as JSON to Dashboard /ingest/edge
end if

Algorithm 2 Cloud Inference Workflow

Require: Frame F from Edge device
Ensure: Cloud detections D_{cloud}
Decode JPEG F
$D_{\text{cloud}} \leftarrow \mathcal{C}(F)$
overlay ← draw bounding boxes on F
Save overlay to server storage
Construct JSON payload {device_id, frame_id, ts, D_{cloud}, overlay_url}
Send payload to Dashboard endpoint /ingest/cloud

The confidence thresholds were determined using the grid-search optimisation procedure outlined in Section 2.4.1. The resulting decision policy follows a selective escalation strategy, whereby the processing pathway is determined according to the confidence level associated with each prediction:

- Low-confidence observations ($< \tau_1$) are discarded to avoid unnecessary data transmission and conserve available network bandwidth.
- Ambiguous observations ($[\tau_1, \tau_2)$) are forwarded to the cloud-based detector C, where the greater computational capacity of the deployed model enables more reliable assessment and reduces the likelihood of incorrect decisions.
- High-confidence observations ($\geq \tau_2$) are finalised directly on the edge device, thereby reducing processing latency and minimising network utilisation.

Table 2 provides a systematic correspondence between the mathematical symbols employed in the proposed framework, their respective definitions and the equations in which they are used.

Table 2
Mapping of Symbols to their Definitions and Corresponding Equations

Symbol	Definition
F_t	RGB frame at time t with dimensions H × W × 3
$\mathcal{E}$	Edge detector (YOLOv8n, RKNN-optimised)
N_t	Number of detections returned by the edge detector for a frame
b_i	Bounding box coordinates of detection i
c_i	Class label of detection i
p_i	Confidence score of detection i, in the range 0 to 1
$\hat{p}_t$	Maximum confidence among detections in a frame
τ_1, τ_2	Confidence thresholds (0.50 and 0.80)
$\mathcal{C}$	Cloud detector (YOLOv8l, FP32)
d_t	Routing decision: DROP, OFFLOAD, or ACCEPT_EDGE
L_E	Edge inference latency
L_C	Cloud inference latency
L_t	End-to-end latency for a frame
N_{edge}	Number of frames processed at the edge
N_{cloud}	Number of frames processed in the cloud
N_{drop}	Number of dropped frames
$\bar{S}_{\text{JPEG}}$	Average compressed JPEG frame size
$B^{(\text{cloud})}$	Bandwidth consumption in cloud-only mode
$B^{(\text{hyb})}$	Bandwidth consumption in hybrid mode
Acc	Accuracy: (TP+TN)/(TP+FP+TN+FN) × 100%

Assuming approximately stationary operating conditions, the expected bandwidth consumption per unit time can be estimated from the probability of cloud offloading and the mean compressed image size as follows:

$$\mathbb{E}[B] \approx P(\tau_1 \leq \hat{p}_t < \tau_2) \cdot \bar{S}_{\text{JPEG}} \tag{7}$$

where $\bar{S}_{\text{JPEG}}$denotes the average size of a compressed image frame.

To evaluate the robustness of the proposed confidence-driven decision mechanism, a threshold sensitivity and ablation analysis was performed using different combinations of lower and upper confidence boundaries. Specifically, the lower threshold was examined at 0.30, 0.40, 0.50 and 0.60, whereas the upper threshold was assessed at 0.70, 0.80 and 0.90, with all configurations satisfying the constraint $\tau_1 < \tau_2$. Each admissible threshold combination was tested using an identical independent orchard test dataset and deployment configuration. The evaluation considered detection accuracy, cloud offloading frequency, rejection rate, end-to-end latency and network bandwidth consumption to establish the influence of threshold selection on overall system performance.

The experimental observations revealed that reducing the lower confidence threshold increased the proportion of predictions accepted locally, although this also raised the possibility of retaining uncertain detections at the edge. Conversely, increasing the lower threshold resulted in more predictions being escalated to cloud-based inference, consequently increasing communication requirements. A similar relationship was observed for the upper confidence boundary. Lower upper-threshold values restricted cloud utilisation but permitted a greater proportion of borderline predictions to be processed locally. In contrast, increasing this threshold resulted in more frequent cloud verification and greater processing latency, while providing additional computational support for uncertain observations.

Among the configurations investigated, the combination of $\tau_1 = 0.50$and $\tau_2 =$

0.80demonstrated the most favourable overall operational balance. This configuration maintained disease detection capability while restricting unnecessary cloud communication and excessive rejection of observations. The experimental evidence therefore provides empirical justification for the selected threshold values, demonstrating that they were established through systematic evaluation rather than arbitrary parameter assignment. Furthermore, the findings illustrate the controllability of the proposed adaptive decision mechanism, whereby confidence thresholds govern the balance among detection reliability, computational expenditure, communication requirements and response latency.

Complexity Analysis

The computational and communication complexity of the proposed hybrid algorithm can be evaluated by independently examining its edge-processing and cloud-processing stages.

Edge-Side Inference

The edge computing platform executes the lightweight detection model E, based on the INT8-quantised YOLOv8n architecture, with computational complexity expressed as:

$$T_E = O(N \cdot k^2 \cdot C) \tag{8}$$

where N represents the number of convolutional layers, $k$denotes the convolutional kernel size, and C corresponds to the number of channels. The application of INT8 quantisation, combined with hardware acceleration through the NPU, reduces the effective computational cost associated with individual processing layers and facilitates real-time execution. Consequently, the system achieves an inference throughput of approximately 20 FPS on the LuckFox Pico Ultra W. Furthermore, the supplementary operations, including confidence maximisation and threshold comparison, introduce only constant-time computational overhead, with a complexity of $O(1)$.

Cloud Offloading

For observations assigned to the cloud-offloading pathway, the edge device transmits a compressed JPEG image with a data size of Sbytes. Accordingly, the associated network transmission complexity is expressed as:

$$T_{\text{net}} = \Theta(S/B) \tag{9}$$

where B denotes the available Wi-Fi uplink bandwidth. Upon receiving the transmitted image, the cloud executes the computationally intensive detection model C, implemented using the FP32-precision YOLOv8l architecture, with computational complexity defined as:

$$T_C = O(N' \cdot k^2 \cdot C') \tag{10}$$

Here, N'represents the number of convolutional layers, $k$denotes the convolutional kernel size, and C'indicates the number of channels in the cloud-based model. Although the computational requirements of Cexceed those of the lightweight edge detector E, its execution cost remains bounded by the available laptop GPU and CPU resources. Additionally, post-processing operations, including the generation of visual overlays and construction of JSON responses, exhibit linear computational complexity with respect to the number of detected objects, M_t, resulting in an overall complexity of $O(M_t)$.

Overall Hybrid Complexity

The expected computational cost per frame is determined by the probability-weighted

contributions of the three possible routing decisions, as formulated below:

$$\begin{aligned}\mathbb{E}[T] = {} & P(\hat{p}_t < \tau_1) \cdot T_{\text{drop}} \\ & + P(\tau_1 \le \hat{p}_t < \tau_2) \cdot (T_E + T_{\text{net}} + T_C) \\ & + P(\hat{p}_t \ge \tau_2) \cdot \left(T_E + T_{\text{json}}\right)\end{aligned} \tag{11}$$

where $T_{\text{drop}} \approx T_E$ represents the computational cost associated with processing and subsequently discarding a frame, whereas $T_{\text{json}} = O(M_t)$denotes the complexity of generating the corresponding JSON output. Given that $T_E \ll T_C$, and that network transmission time, T_{net}, constitutes the dominant cost under bandwidth-constrained conditions, the proposed hybrid architecture limits the computationally expensive cloud inference and associated data transmission to observations classified as ambiguous. Consequently, the selective offloading mechanism reduces unnecessary cloud utilisation while retaining the computational capacity required for further assessment of uncertain predictions.

Memory and Storage

At the edge level, memory consumption is constrained by the dimensions of the input tensor, $H \times W \times 3$, together with the intermediate activation maps generated by model E. Both memory requirements remain constant for a fixed network architecture and input resolution. In the cloud environment, GPU memory utilisation is primarily determined by the architectural dimensions of model C, particularly $N'$and C'; nevertheless, these requirements remain within the capacity of conventional computing hardware. Furthermore, when optional retraining buffers are employed, their memory requirements increase linearly with the number of offloaded frames retained for subsequent processing, exhibiting a complexity of $O(T)$, where Tdenotes the number of stored time steps.

2.3 Experimental Setup

System-level performance indicators, including inference latency and bandwidth consumption, were measured using unlabelled live data, whereas detection accuracy was assessed independently using a held-out annotated test dataset. During live operation, confidence-based monitoring and selective escalation to cloud-based inference provided an indirect mechanism for maintaining prediction reliability in the absence of ground-truth annotations. To enable a consistent and label-based comparison of cloud-only, edge-only and hybrid execution, a separate collection of annotated orchard photographs was established and reserved exclusively for testing, without overlap with the training or validation datasets. Each image contained zero or more apple disease instances, accompanied by corresponding bounding-box annotations and class labels.

The experimental testbed consisted of three interconnected components communicating through Wi-Fi 6 (IEEE 802.11ax):

- Edge Device: The edge platform comprised a LuckFox Pico Ultra W equipped with a Rockchip RV1106 NPU (approximately 1 TOPS INT8) and 512MB DDR3 memory. A quantised YOLOv8n model, implemented in INT8 precision using the RKNN framework, served as the edge detection model E. The dual-threshold decision mechanism was implemented in C++, with FastAPI endpoints supporting telemetry transmission and detection result ingestion.
- Cloud Offload Server: An Apple MacBook Air M3, featuring an 8C CPU, 10C GPU and 16GB RAM, functioned as the cloud inference server. A FastAPI-based service hosted the YOLOv8l model operating at FP32 precision and provided the /infer endpoint for processing images received from the edge device.
- Client Dashboard: A separate laptop running FastAPI served as the client dashboard. It received

detection results through the /ingest/edge and /ingest/cloud endpoints, generated visual overlays and maintained records of the corresponding performance measurements.

- Three independent measurement configurations were implemented using the same collection of images:
- Cloud-Only: Each image was transmitted from the edge device to the cloud server, where disease detection was performed. The resulting predictions and associated performance measurements were subsequently returned to the dashboard.
- Edge-Only: All images were processed locally using the edge detection model, after which the resulting predictions and performance measurements were transmitted directly to the dashboard. No requests were generated for cloud-based inference.
- Hybrid (Dual-Threshold): The edge model first calculated the frame-level confidence score, $\hat{p}$, defined as the maximum prediction confidence associated with each image. Images satisfying $\hat{p} \geq 0.80$were processed and accepted locally, with their detection results forwarded directly to the dashboard (ACCEPT_EDGE). Images for which $0.50 \leq \hat{p} < 0.80$were transferred to the cloud for further inference, and the resulting predictions were returned to the dashboard (OFFLOAD). Conversely, images with $\hat{p} < 0.50$were discarded (DROP) and recorded on the dashboard as instances of "no detection".

During the hybrid evaluation, the routing decision for every image $i \in \{1, \ldots, 45\}$was recorded using the corresponding decision category $r_i \in \{\text{ACCEPT_EDGE}, \text{OFFLOAD}, \text{DROP}\}$. The resulting observations were subsequently aggregated to determine the total number of images processed locally, transferred to the cloud or rejected, represented by N_{edge}, $N_{\text{cloud}}$and N_{drop}, respectively. These quantities satisfied the following relationship:

$$N_{\text{edge}} + N_{\text{cloud}} + N_{\text{drop}} = 45.$$

All timing measurements were obtained using timestamps recorded at individual communication and processing stages, while HTTP-layer payload sizes were monitored to quantify network bandwidth consumption. In addition, system-level performance indicators were collected from unlabelled live traffic to characterise the operational behaviour of the proposed architecture under real-time conditions. Nevertheless, accuracy calculations for all three execution configurations were based exclusively on the annotated image dataset, ensuring a consistent and comparable evaluation.

Confidence Threshold Optimization

A systematic grid-search strategy was employed to determine the most appropriate values for the two confidence thresholds governing the proposed routing mechanism. Candidate lower-threshold values (τ_1) were selected from 0.30, 0.40, 0.50 and 0.60, whereas the upper threshold (τ_2) was varied across 0.70, 0.80 and 0.90. Only threshold combinations satisfying the constraint $\tau_1 < \tau_2$were considered valid for experimental evaluation. Each admissible combination was assessed using the same independent orchard test dataset while maintaining identical model configurations, hardware specifications and network conditions. For every image, the corresponding ACCEPT_EDGE, OFFLOAD or DROP decision was recorded to establish the operational behaviour of the candidate routing policy.

The performance of each threshold combination was quantified using detection accuracy, cloud offloading rate, rejection rate, end-to-end latency and network bandwidth consumption. Threshold determination was treated as a multi-criteria optimisation problem because improvements in detection reliability may introduce additional computational expenditure and communication requirements. Accordingly, candidate configurations were evaluated collectively in terms of

detection performance, response latency, network utilisation and cloud resource consumption. The final threshold combination was selected according to its overall performance across these interdependent criteria. The optimisation results identified $\tau_1 = 0.50$and $\tau_2 = 0.80$as the most suitable configuration for the proposed system. This systematic evaluation establishes an empirical justification for the selected decision boundaries and reduces reliance on manually specified threshold values.

Statistical Validation

A statistical evaluation was undertaken to examine the consistency and significance of performance differences among the edge-only, cloud-only and hybrid execution configurations. Eight repeated experimental trials were conducted under equivalent hardware, network, model and dataset conditions, with an identical sequence of images employed throughout all trials. Detection accuracy, end-to-end latency and bandwidth consumption were recorded during each experimental repetition. The resulting mean performance values were accompanied by 95% confidence intervals to characterise the variability associated with repeated measurements.

The measured mean end-to-end latency was 0.21 ± 0.01 s (95% CI) for edge-only execution, 0.86 ± 0.03 s for cloud-only processing and 0.24 ± 0.01 s for hybrid inference. Correspondingly, mean bandwidth consumption was 0.00 ± 0.00 MB, 18.42 ± 0.31 MB and 4.07 ± 0.18 MB, respectively. The distributional characteristics of the experimental measurements were examined using the Shapiro-Wilk test to determine whether the normality assumption was satisfied. Depending on the distribution of the paired differences, either the paired t-test was employed when normality was established or the Wilcoxon signed-rank test was used otherwise. All statistical comparisons were performed using a two-sided significance level of 0.05. TAhe hybrid execution configuration demonstrated a statistically significant reduction in latency relative to cloud-only inference ($p < 0.001$, Cohen's $d = 2.41$). A similarly significant reduction was observed in bandwidth consumption ($p < 0.001$, Cohen's $d = 3.17$). To maintain consistency in the accuracy assessment, all three execution configurations were evaluated using the same independent annotated test dataset. The combination of confidence intervals, paired statistical comparisons and effect-size estimates provided a quantitative basis for determining the robustness of the observed performance differences across repeated experimental trials.

2.4 Evaluation Metrics

The three execution configurations were evaluated using an identical set of annotated images to ensure consistency and comparability across experimental conditions. For each configuration, performance was assessed using three principal metrics: detection accuracy, inference latency and network bandwidth consumption.

Accuracy (%)

For every test image, the predicted bounding boxes were matched against the corresponding ground-truth annotations using an intersection-over-union (IoU) threshold of ≥0.5. The resulting matches were used to construct the confusion matrix and calculate class-agnostic detection performance metrics. Overall detection accuracy was calculated using the following expression:

$$Acc = \frac{TP + TN}{TP + FP + TN + FN} \times 100\% \qquad (12)$$

An identical set of ground-truth annotations was maintained across the edge-only, cloud-only and

hybrid execution configurations to ensure a consistent basis for performance comparison. Within the hybrid configuration, images assigned to the DROP category were treated as producing a "no detection" prediction. Accordingly, such cases were counted as true negatives (TN) when the corresponding ground-truth image contained no disease instances, whereas images containing disease instances were classified as false negatives (FN).

Latency (s)

The end-to-end latency for each test image, denoted by L_i, was determined by measuring the elapsed time between the initiation of image processing at the edge device and the subsequent rendering of the corresponding annotated detection result on the dashboard. For the edge-only configuration, the latency was calculated as the sum of the local inference duration and the time required to deliver and display the resulting detection output:

$$L_i^{(\text{edge})} = \left(T_E^{\text{end}} - T_E^{\text{start}}\right) + \left(T_D^{\text{disp}} - T_E^{\text{end}}\right) \qquad (13)$$

For cloud-only execution, the overall latency incorporated the image transmission time from the edge to the cloud, the cloud response interval, the cloud inference duration, the subsequent transfer of detection results to the dashboard and the final rendering time. Accordingly, it was determined using:

$$\begin{aligned} L_i^{(\text{cloud})} = &\left(T_{EC}^{\text{send}} - T_E^{\text{start}}\right) + \left(T_{EC}^{\text{recv}} - T_{EC}^{\text{send}}\right) \\ &+\left(T_C^{\text{end}} - T_C^{\text{start}}\right) \\ &+\left(T_{CD}^{\text{recv}} - T_{CD}^{\text{send}}\right) + \left(T_D^{\text{disp}} - T_{CD}^{\text{recv}}\right) \end{aligned} \qquad (14)$$

For the hybrid configuration, the latency calculation depended on the routing decision assigned to each image. Images classified as ACCEPT_EDGE were evaluated using $L_i^{(\text{edge})}$, whereas those assigned to OFFLOAD were assessed using $L_i^{(\text{cloud})}$. For images classified as DROP, the corresponding latency comprised the time required to make the local rejection decision together with the time taken to communicate and display the rejection status on the dashboard.

To provide a comprehensive assessment of temporal performance, the mean and median latency values were calculated across the test images, accompanied by their corresponding 95% confidence intervals.

Bandwidth (Mbps)

Network communication overhead was quantified by aggregating the sizes of transmitted payloads recorded at the HTTP layer. For the cloud-only configuration, the total bandwidth consumption was calculated by summing the sizes of all JPEG images transmitted from the edge device to the cloud server:

$$B^{(\text{cloud})} = \sum_{i=1}^{100} \text{size_bytes}\left(\text{Edge} \rightarrow \text{Cloud JPEG}_i\right) \qquad (15)$$

For the hybrid configuration, bandwidth consumption was determined by considering only those images selected for cloud-based processing. Accordingly, the total JPEG transmission volume was calculated as:

$$B^{(\mathrm{hyb})} = \sum_{i \in O} \mathrm{size_bytes}\,(\mathrm{Edge} \rightarrow \mathrm{Cloud\ JPEG}_i) \qquad (16)$$

where O denotes the index set containing all images transferred to the cloud for additional inference. JSON telemetry transmitted from the edge device to the dashboard and from the cloud server to the dashboard was quantified separately and documented to provide a complete account of communication overhead. In the edge-only configuration, JPEG image transmission to the cloud was not performed; consequently, the corresponding uplink bandwidth consumption was zero.

2.5 Dataset

The image dataset was acquired from the Issyk Aport apple orchard during 14 to 15 June 2025 using a professional RGB camera. A sufficient number of images was initially collected, followed by a systematic data-cleaning procedure to ensure their suitability for subsequent analysis. The annotation process was conducted in collaboration with biologists from the National Agrarian Scientific and Educational Center of Kazakhstan, with particular emphasis on four consolidated disease categories: fungal diseases (apple scab and powdery mildew), bacterial diseases (fire blight), physiological disorders (nutrient deficiency) and pest-related damage (miner moth and aphid damage).

A domain-specific apple disease dataset was established using images captured under realistic orchard conditions, encompassing both healthy and diseased leaf and fruit samples. The collected images reflect naturally occurring variations in illumination, background complexity, object orientation and occlusion, as well as differences in disease manifestation. Each image underwent manual examination and annotation by professional biologists, who identified disease instances using bounding boxes and assigned the corresponding class labels. To maintain annotation reliability, consistency was examined before model development, and observations involving ambiguous disease characteristics were subjected to additional review before being incorporated into the dataset.

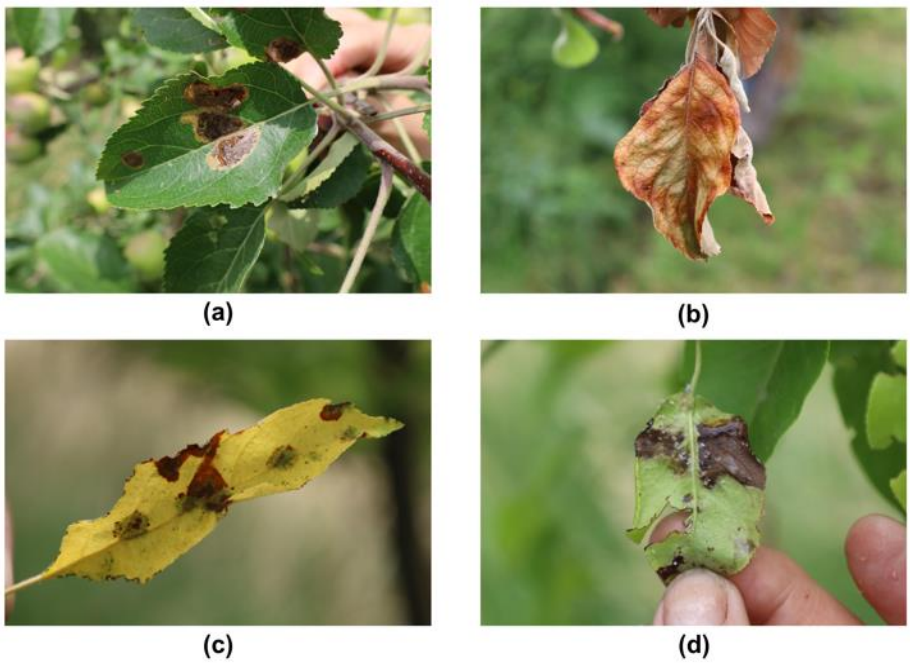


Fig.3: Example of Disease Classes. (a) Fungal, (b) Bacterial, (c) Physio, (d) Pest

Following annotation and quality assessment, the dataset was partitioned into training, validation and test subsets. The test subset was maintained independently throughout model development to prevent its involvement in the training and validation processes. Class distributions were also examined to identify potential imbalances among the disease categories. Furthermore, data augmentation procedures were restricted exclusively to the training subset to preserve the integrity of the validation and test data. The resulting dataset served as the common experimental foundation for all investigations conducted in this study. Nevertheless, an expanded dataset is currently under development to incorporate a broader range of environmental conditions and improve the representativeness of orchard observations. Formal inter-annotator agreement was not calculated as part of the present investigation; this aspect is recognised as a limitation and will be addressed in subsequent stages of dataset development.

3. RESULTS

Image annotation was carried out using the Roboflow platform. Initial dataset analysis revealed a substantial class imbalance, with the number of instances belonging to the fungal and bacterial disease categories being approximately 5 to 6 times higher than those representing pest damage and physiological disorders. To mitigate this imbalance, targeted data augmentation was applied exclusively to the minority categories, namely pest damage and physiological disorders, within the training subset. The augmentation procedures incorporated affine and perspective transformations, image blurring, brightness and contrast adjustments, colour shifting, Contrast Limited Adaptive Histogram Equalisation (CLAHE), JPEG compression and coarse dropout. These techniques increased the representation of underrepresented categories, resulting in a more balanced class distribution while preserving the original, unaugmented composition of the validation and test subsets. Figure 4 presents the class distributions before and after augmentation, illustrating the effectiveness of the targeted augmentation strategy in addressing the imbalance among minority classes.

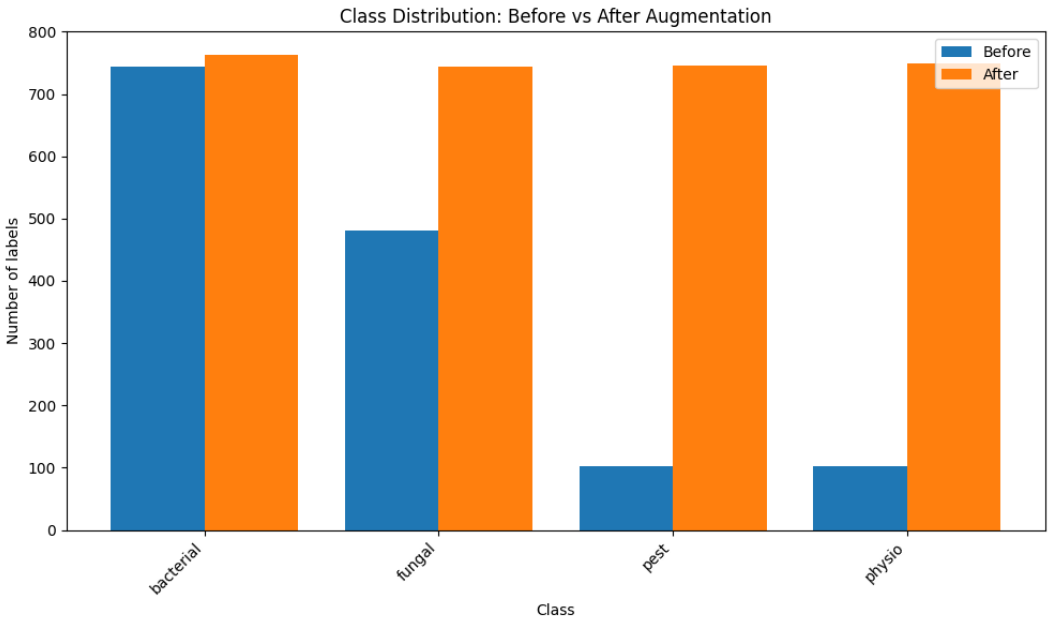


Fig.4: Class Balancing after Augmentation

Table 3 presents a comparative overview of the class distributions before and after the application of data augmentation. The bacterial category exhibits a relatively modest increase, rising

from 744 to 762 instances (2.42%), whereas the fungal category expands from 480 to 744 instances (55.00%). More substantial augmentation is applied to the two minority categories. Specifically, the number of pest-related instances increases from 103 to 746 (624.27%), while the physiological category grows from 103 to 749 instances (627.18%). Following augmentation, the representation of all four categories falls within a relatively narrow range of approximately 744 to 762 instances. This adjustment produces a substantially more balanced training dataset, thereby mitigating class imbalance during model optimisation.

Table 3
Class Balancing

ClassID	ClassName	Before	After	Delta	Change (%)
0	Bacterial	744	762	18	2.42
1	Fungal	480	744	264	55.00
2	Pest	103	746	643	624.27
3	Physio	103	749	646	627.18

The completed dataset was partitioned into three subsets, comprising 80% for model training, 10% for validation and the remaining 10% for independent testing.

3.1 Model Selection

Table 4 presents a comparative assessment of representative object detection architectures in terms of computational complexity, inference speed and detection accuracy. The findings highlight the differing model capacity requirements associated with efficient edge deployment and high detection performance, indicating that these objectives cannot always be achieved simultaneously using a single model configuration. In this regard, a hybrid edge-cloud architecture offers a practical means of balancing these competing requirements. The comparative results provide insight into the suitability of different object detection models for deployment across edge and cloud environments. In particular, YOLOv8n and YOLOv8l demonstrate a favourable combination of detection accuracy, computational requirements and inference speed, supporting their selection as complementary detection models within the proposed hybrid edge-cloud architecture.

Table 4
Comparison of Edge and Cloud Object Detection Models

Model	Params (M)	FLOPs (B)	FPS	Board	mAP@0.5:0.95
YOLOv8n-int8 (Proposed)	3.2	8.9	20	RV1106 NPU 1 TOPS	58%
YOLOv4-tiny [20]	4.0	7.0	76	Coral Dev Board	21%
SSD MobileNet V2 [20]	5.0	1.3	99	Jetson Xavier NX (GPU)	29%
YOLOv5s [21]	7.2	16.5	30	Jetson Xavier NX (15W mode)	37%
EfficientDet-Lite0 [22]	3.9	2.5	12	Raspberry Pi 4 + RKNN (2 TOPS)	22%
YOLOv7-Tiny (QAT) [22]	6.2	13	2	Raspberry Pi 4 + Coral USB TPU	35%
YOLOv8l-FP32 (Proposed)	43	165.2	418	A100 GPU	52.9%
EfficientDet-D7 [23]	52-77	325-410	31	V100 GPU	52.2%
Swin-L + Cascade RCNN [24]	200+	1000+	10-20	A100 GPU	55-58%

The selection of suitable detection architectures for hybrid edge-cloud deployment necessitates a careful consideration of detection accuracy, computational complexity and the feasibility of real-time execution on resource-constrained hardware. Previous investigations have established the widespread use of lightweight detection architectures, including YOLOv4-tiny, SSD MobileNet V2 and EfficientDet-Lite variants, for embedded inference. Conversely, larger architectures, such as EfficientDet-D7 and Swin-L + Cascade R-CNN, generally achieve superior performance in cloud-based

benchmarks, albeit at the expense of substantially greater computational and memory requirements.

To evaluate the suitability of lightweight architectures for edge deployment, the proposed YOLOv8n-INT8 model was benchmarked on the Rockchip RV1106 NPU against performance results reported in previous studies. Kang and Somtham [20] investigated YOLOv4-tiny and SSD MobileNet V2 using the Coral Dev Board and Jetson Xavier NX platforms, reporting exceptionally high inference rates of 76 and 99 FPS, respectively. However, their detection accuracy remained relatively low, with mAP@0.5:0.95 values of ≤29%, primarily owing to quantisation-related limitations and activation mismatches on the EdgeTPU. Galliera and Suri [21] evaluated YOLOv5s on the Jetson Xavier NX, achieving 30 FPS and 37% mAP, thereby demonstrating a more favourable compromise between computational efficiency and detection performance. Similarly, Zagitov et al. [22] assessed EfficientDet-Lite0 and YOLOv7-Tiny using Raspberry Pi 4 accelerators, obtaining inference speeds of only 2 to 12 FPS and detection accuracy below 35%, which illustrates the computational limitations encountered when processing images at higher resolutions.

In comparison, the proposed YOLOv8n-INT8 architecture achieved an inference rate of 20 FPS and 58% mAP on a domain-specific orchard dataset. These results demonstrate that the integration of quantisation-aware optimisation with the RKNN deployment toolchain can maintain comparatively high detection accuracy even when operating on ultra-low-cost neural processing hardware. Figure 5 depicts the complete model deployment workflow. Initially, the baseline YOLOv8n architecture is fine-tuned using the custom orchard dataset and subsequently exported in ONNX format. The resulting optimised model is then quantised into the RKNN INT8 format, enabling its deployment on the LuckFox Pico Ultra embedded edge platform.

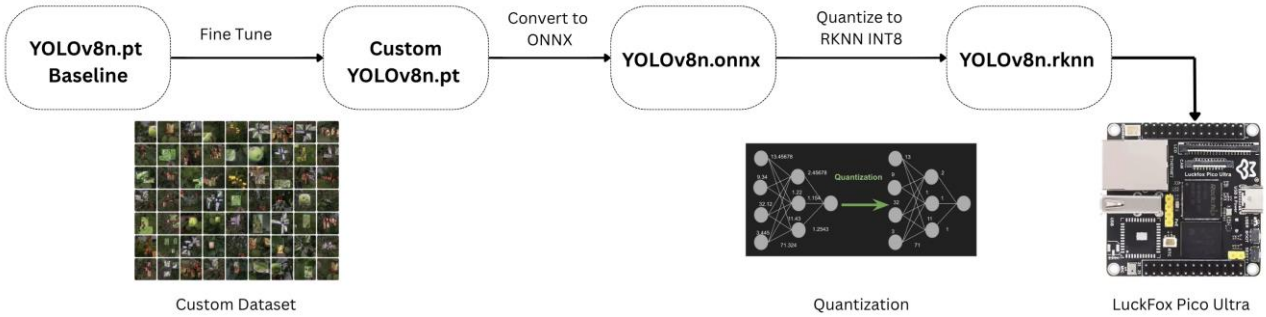


Fig.5: Model Pipeline

For cloud-based processing, the primary optimisation objective shifts from minimising computational expenditure per watt towards maximising inference throughput while maintaining high detection accuracy. The proposed YOLOv8l-FP32 model achieved 52.9% mAP@0.5:0.95 and an inference throughput of 418 FPS on an NVIDIA A100 using TensorRT acceleration, consistent with the benchmarks reported by Ultralytics [25]. In comparison, EfficientDet-D7 [23] achieves a comparable accuracy of 52.2%, but requires substantially greater computational resources, with a complexity of 325 to 410 GFLOPs and an inference speed of approximately 30 FPS on a V100 GPU. Similarly, the Swin-L + Cascade R-CNN architecture [24] achieves an AP of up to 58%, although this performance requires more than 1000 GFLOPs and is accompanied by lower inference throughput. Relative to these computationally intensive architectures, the proposed YOLOv8l configuration provides a more favourable relationship between average precision and computational requirements, making it a suitable candidate for large-scale streaming applications.

The comparative results for the selected edge and cloud detection architectures are summarised in the corresponding table. It should be recognised that the reported performance values are not

directly normalised across datasets, as the evaluated baseline models were assessed using different benchmarks, including COCO, VisDrone and independently developed datasets. Nevertheless, the overall performance trends remain informative. Conventional lightweight edge architectures frequently sacrifice detection accuracy to achieve higher inference speeds, whereas the proposed YOLOv8n-INT8 maintains comparatively strong detection performance while delivering an acceptable inference rate on the RV1106 NPU. On the cloud side, YOLOv8l provides a favourable combination of detection accuracy, processing throughput and computational efficiency compared with substantially larger architectures. Selecting these complementary models enables the proposed hybrid architecture to exploit the respective advantages of local and cloud-based inference.

To facilitate practical deployment of the selected detection models, a dedicated conversion and optimisation pipeline was developed according to the hardware requirements of the Rockchip RV1106 NPU integrated into the LuckFox Pico Ultra W. The procedure initially employs the baseline YOLOv8n.pt architecture, pre-trained on the COCO dataset. This model is subsequently fine-tuned using the custom apple disease dataset to generate a domain-adapted YOLOv8n.pt checkpoint with improved suitability for orchard environments. Following completion of the training process, the resulting PyTorch checkpoint is exported to the ONNX format (YOLOv8n.onnx), which serves as an intermediate representation independent of the target hardware architecture. The ONNX model is then converted into the Rockchip Neural Network (RKNN) format (YOLOv8n.rknn) using the RKNN toolkit. This sequential conversion and optimisation procedure is designed to maintain detection performance following quantisation while enabling real-time inference within the stringent computational and power limitations of embedded devices deployed in agricultural environments. Figure illustrates the complete processing workflow, extending from baseline model training to final deployment on the NPU.

Model fine-tuning was performed using Google Colab Pro with an NVIDIA A100 GPU. The training configuration comprised 100 epochs, an input image resolution of 640×640 and a batch size of 8. To promote stable convergence and mitigate overfitting, a cosine learning rate scheduling strategy was employed alongside early stopping with a patience of 50 epochs. Table 5 provides a summary of the hyperparameters and experimental settings adopted during YOLOv8n fine-tuning. Specifically, the model was trained on GPU hardware for 100 epochs using images with a resolution of 640 × 640 and a batch size of 8. The combination of cosine learning rate scheduling and early stopping with a patience of 50 epochs was incorporated to improve training convergence and limit overfitting.

Table 5
Training Hyperparameters for YOLOv8n Fine-Tuning

Hyperparameter	Value
Epochs	100
Image size	640
Batch size	8
Device	0 (GPU)
Early stopping patience	50
Cosine learning rate schedule	Enabled

Training progression was assessed by tracking the evolution of the loss components, including bounding-box, objectness and classification losses, together with precision, recall and mAP@0.5:0.95 throughout the training epochs. The resulting performance trajectories are presented in Figure 6, which illustrates the training and validation behaviour of the YOLOv8n and YOLOv8l architectures over 100 epochs. The loss curves exhibit a consistent downward trend, whereas precision, recall,

mAP@0.5 and mAP@0.5:0.95 progressively increase before approaching convergence. These trends indicate stable optimisation, effective feature learning and satisfactory convergence behaviour for both model architectures.

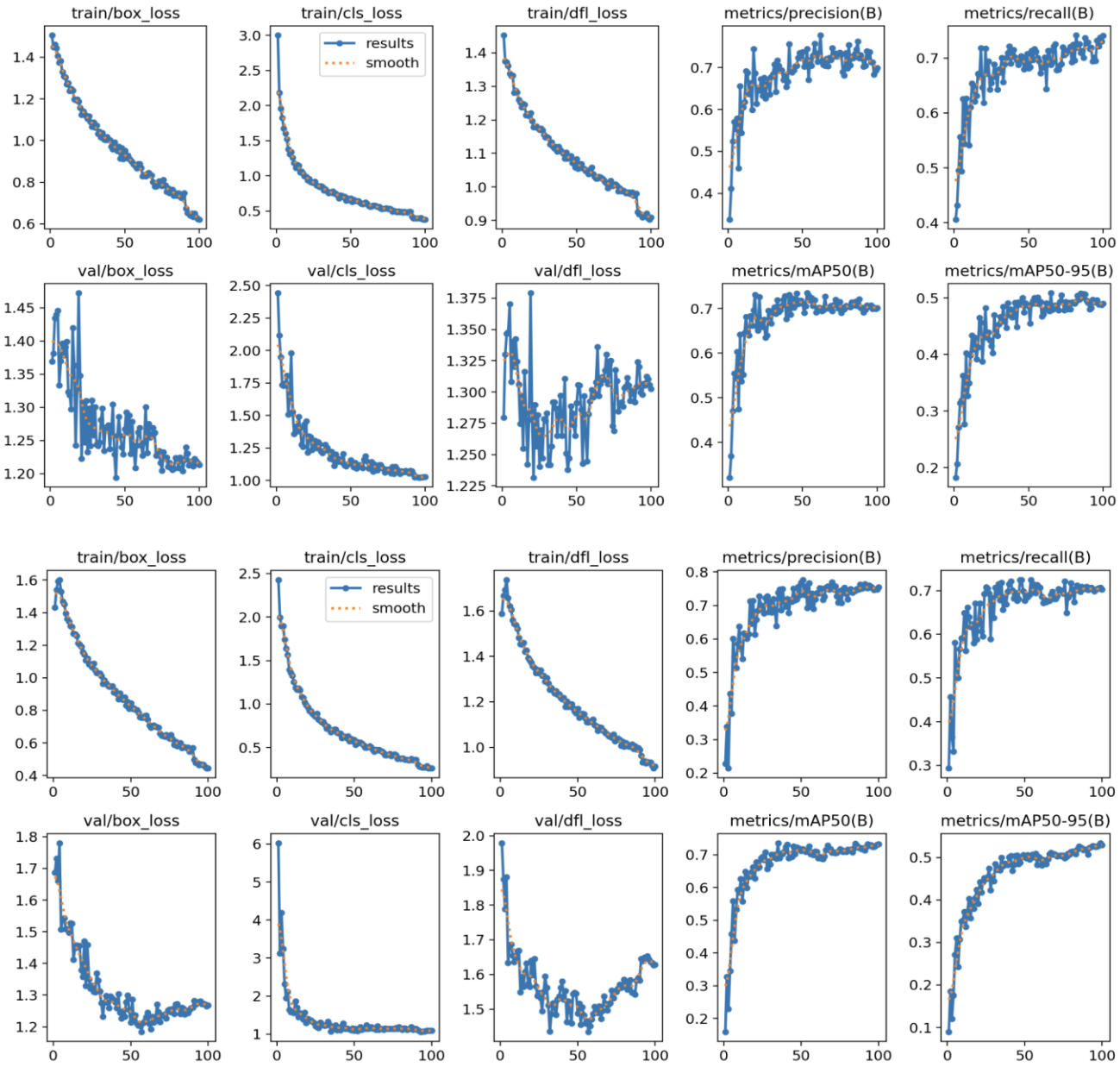


Fig.6: Training Metrics for YOLOv8n and YOLOv8l

Table 6 provides a comparative evaluation of the three deployment configurations in terms of their overall performance. The hybrid architecture delivers the highest detection accuracy (81.6%), while simultaneously achieving substantial reductions in end-to-end latency and bandwidth consumption relative to cloud-only processing. Although the edge-only configuration offers the shortest response time, its detection accuracy is comparatively lower. Conversely, cloud-only execution improves detection accuracy but incurs considerably greater network latency and communication overhead. Collectively, these findings demonstrate that the proposed hybrid strategy achieves the most favourable compromise among detection accuracy, response time and communication efficiency.

Table 6
Measured Performance

Scheme	Accuracy (%)	Inference Latency (s)	Network Latency (s)	Total Latency (s)	BW (Mbps)
Edge-only	75.6	0.231	0.000	0.231	0.0
Cloud-only	80.6	0.285	6.256	6.541	62.8
Hybrid	81.6	0.225	1.376	1.601	40.7

The threshold optimisation analysis established $\tau_1 = 0.50$ and $\tau_2 = 0.80$ as the most effective combination among the evaluated routing configurations. This threshold pair achieved a favourable trade-off across detection accuracy, cloud offloading frequency, rejection rate, end-to-end latency and bandwidth consumption. Configurations employing lower threshold values resulted in a greater proportion of locally accepted predictions, but also increased the likelihood of retaining uncertain observations at the edge. Conversely, higher threshold settings promoted more frequent cloud-based verification, thereby increasing communication requirements and associated resource consumption. The selected thresholds consequently represent a balanced compromise between prediction reliability and computational resource utilisation. On the basis of these experimental findings, $\tau_1 = 0.50$and $\tau_2 = 0.80$ were established as the final parameters governing the proposed adaptive routing policy.

3.2 Baselines

Previous investigations into apple disease detection have frequently adopted lightweight YOLO-based architectures to achieve an appropriate compromise between detection accuracy and computational efficiency on resource-constrained hardware. Zou et al. [9] developed LightYOLO-AppleLeafDx using the SCIDB and Kaggle PlantVillage datasets, reporting an mAP@0.5 approaching 95% while retaining computational characteristics suitable for deployment on devices with limited resources. Although their findings demonstrate the effectiveness of dataset-oriented model optimisation under controlled conditions, the experimental setting does not adequately encompass the complexities associated with practical orchard deployment. In a complementary investigation,Wang et al. [8] developed a customised orchard dataset incorporating background complexity and variations in illumination and introduced an enhanced lightweight detection architecture that achieved 94.6% mAP@0.5. Their findings emphasise the importance of assessing detection models under representative field conditions, where resilience to environmental variations is an essential consideration alongside detection accuracy. Collectively, these studies provide useful reference points for evaluating the relationship between computational efficiency and environmental robustness. Nevertheless, they also reveal an important research limitation, as systematic investigations of hybrid approaches integrating the computational advantages of edge processing with the enhanced detection capabilities of cloud infrastructure remain relatively scarce in agricultural disease monitoring.

The experimental evaluation was conducted using a dedicated, annotated orchard dataset designed to represent typical field conditions. Three distinct deployment configurations were investigated to establish their comparative performance. The edge-only configuration employed the INT8-quantised YOLOv8n model, whereas the cloud-only configuration utilised the FP32 YOLOv8l architecture. The third configuration implemented the proposed adaptive decision-support mechanism, which dynamically selected the inference pathway according to prediction confidence. Under this policy, predictions exceeding the upper confidence threshold were accepted locally, observations with intermediate confidence were transferred to the cloud for additional assessment, and predictions below the lower threshold were rejected. This confidence-driven routing mechanism

enabled the system to adapt its computational strategy to the reliability of individual predictions.

The experimental results demonstrated detection accuracies of 75.6% for edge-only inference, 80.6% for cloud-only inference and 81.6% for the proposed hybrid approach. Approximately 22.2% of the evaluated images were escalated to the cloud, while only 2.2% were rejected. Consequently, nearly 78% of the observations were processed locally without requiring additional cloud inference. Relative to edge-only execution, the proposed approach improved detection accuracy by 6%. Furthermore, bandwidth consumption was reduced by 35% compared with cloud-only processing, while end-to-end latency was reduced by a factor of 4.1. The average latency of the hybrid configuration was 1.60 s, compared with 6.54 s for cloud-only inference. These findings illustrate the effectiveness of the proposed adaptive decision-support mechanism in simultaneously addressing multiple system-level performance objectives.

Detection accuracy remained comparable to that of cloud-based processing, while response latency approached the performance characteristics of local inference. At the same time, selective cloud utilisation substantially reduced network communication requirements. Rather than distributing computational workloads uniformly, the proposed framework allocated processing resources according to prediction confidence, reserving cloud-based inference for observations requiring additional verification. Consequently, the majority of processing decisions were completed locally, reducing dependence on continuous network communication. Overall, the experimental findings demonstrate the potential of confidence-guided resource allocation to improve computational efficiency and support scalable orchard disease monitoring, particularly in environments characterised by limited network connectivity.

4. DISCUSSION

The central methodological contribution of this study is the development of an adaptive decision-support mechanism that determines the timing, location and execution pathway of inference according to prediction confidence. This capability transforms the proposed architecture from a conventional object detection pipeline into an intelligent decision-support system in which computational operations are governed by the reliability of individual predictions.

The reliability of the experimental assessment was established through a dedicated annotated orchard dataset designed to reflect representative field conditions. The collected observations incorporated naturally occurring variations in illumination, background complexity, partial occlusion and disease manifestation. Three alternative deployment configurations were investigated: edge-only inference using the INT8-quantised YOLOv8n model on the LuckFox Pico Ultra W, cloud-only inference employing the FP32 YOLOv8l architecture, and the proposed hybrid deployment strategy. The latter incorporated a confidence-driven decision mechanism that assessed each image before determining its final processing pathway. Predictions associated with high confidence were accepted directly at the edge, whereas observations exhibiting intermediate confidence were transferred to the cloud for additional verification. Images associated with exceptionally low confidence were rejected. This arrangement enabled computational resources to be allocated dynamically rather than following a predetermined execution strategy. Consequently, the proposed architecture operated as an intelligent decision-support framework rather than merely performing conventional object detection.

The experimental findings substantiate the effectiveness of this decision-oriented architecture. The edge-only configuration achieved an accuracy of 75.6%, reflecting the performance limitations associated with resource-constrained embedded hardware. Cloud-only inference improved accuracy to 80.6%, although this improvement was accompanied by increased response latency and

substantial bandwidth requirements. In comparison, the proposed hybrid configuration achieved the highest accuracy of 81.6%. Approximately 22.2% of the images required transmission to the cloud, while approximately 2.2% were rejected because their prediction confidence remained below the specified acceptance threshold. Consequently, nearly 78% of the observations were processed locally. These results indicate that the majority of operational decisions could be completed without requiring remote computational resources, allowing cloud infrastructure to be reserved for observations involving greater uncertainty. Computational expenditure was therefore concentrated on cases for which additional analysis was considered necessary.

An important characteristic of the proposed framework is its treatment of prediction confidence as an operational decision variable rather than simply an indicator of model performance. Confidence directly governs resource allocation, determines the location of inference and regulates communication requirements. Through this mechanism, the system automatically selects the appropriate computational pathway according to the reliability of its predictions. Unnecessary cloud requests are consequently minimised, while reliable detection results remain accessible despite the limited computational capabilities of embedded devices. In this respect, the proposed methodology establishes a practical connection between predictive uncertainty and operational decision-making. The decision-support capability also improves the way individual observations are handled. Straightforward cases are resolved immediately through local inference, observations requiring further examination are submitted to the cloud-based detector, and predictions with insufficient reliability are excluded from the decision process. This hierarchical assessment strategy reduces the likelihood of generating inappropriate decisions and reflects a reasoning process in which routine observations receive immediate attention, whereas more challenging cases undergo additional evaluation. Such differentiated processing cannot be readily achieved through conventional fixed edge-only or cloud-only deployment, in which all observations follow the same computational pathway irrespective of their prediction confidence.

The benefits of the proposed architecture extend beyond improvements in detection accuracy. Compared with cloud-only deployment, the hybrid configuration reduced bandwidth consumption by 35% and decreased end-to-end latency by a factor of 4.1. Its response time remained comparatively close to that of edge-only execution, while its detection accuracy was comparable to that achieved through cloud-based inference. These findings demonstrate the simultaneous consideration of several interconnected performance objectives, including detection reliability, response time, communication expenditure and computational resource utilisation. Rather than optimising these objectives independently, the proposed decision mechanism addresses them collectively by selecting the inference location according to prediction confidence. Local processing is prioritised when the available confidence is sufficient, whereas cloud-based analysis is initiated when additional computational evidence is required. Prediction uncertainty is thus translated into actionable operational decisions.

Beyond its computational implications, the proposed decision-support framework has potential relevance to practical orchard management. Timely identification of disease symptoms can facilitate earlier intervention and more informed crop-management decisions. Disease-related risk information may support more targeted pesticide application and the prioritisation of inspection activities according to the distribution of detected symptoms. Furthermore, reducing unnecessary cloud communication can contribute to lower operational and network-related expenditure. These characteristics make the proposed approach particularly relevant to agricultural environments where network connectivity and computational infrastructure are limited. The principal contribution of this investigation therefore resides in the adaptive decision-support mechanism rather than in the

implementation of hybrid computing alone. Although edge-cloud integration provides the underlying computational infrastructure, the methodological innovation concerns the automatic determination of when additional computational assistance is necessary and where the corresponding inference should be executed. By linking resource allocation directly to prediction confidence, the framework establishes a systematic relationship between detection reliability and computational decision-making. The same underlying principle could potentially be extended to other agricultural applications, including pest identification, fruit quality assessment, irrigation management and nutrient deficiency diagnosis. Its integration into autonomous agricultural platforms may also facilitate more scalable and intelligent precision farming operations.

Nevertheless, the experimental investigation was conducted within a specific apple orchard environment, and the collected observations do not encompass the full diversity of agricultural conditions. Although the selected orchard provides a relevant setting for investigating disease detection within Kazakhstan's apple production sector, variations in climatic conditions, illumination, occlusion, background characteristics and disease manifestation across other agricultural regions may influence detection performance. The proposed decision-support mechanism relies on prediction confidence to determine the appropriate inference pathway; therefore, its underlying principle is potentially transferable to other crops and detection architectures. However, such applications require the corresponding crop-specific models to undergo appropriate training and validation before deployment. Similarly, adaptation to alternative embedded platforms requires device-specific calibration to accommodate differences in computational capacity and communication constraints. The confidence thresholds may also require recalibration to reflect different environmental and operational conditions. Accordingly, although the proposed methodology offers potential for broader edge-cloud deployment, its reliability across diverse agricultural environments must be established through further validation. Future research will therefore focus on expanding dataset coverage and systematically evaluating cross-domain generalisation and deployment across different agricultural settings

5. CONCLUSION

This study developed an adaptive decision-support system for automated apple disease detection using a hybrid edge-cloud architecture. A confidence-based inference mechanism enabled high-confidence predictions to be processed locally, uncertain cases to be transferred to the cloud, and low-confidence predictions to be rejected. An INT8-quantised YOLOv8n model deployed on the low-cost LuckFox Pico Ultra W achieved over 20 FPS with an average edge latency of 231 ms. Cloud processing employed YOLOv8l through Wi-Fi, HTTP and FastAPI, with only 22.2% of images requiring additional analysis. The system achieved 81.6% detection accuracy, a 6% improvement over edge-only inference, while reducing bandwidth consumption by 35% and end-to-end latency by a factor of 4.1 compared with cloud-only processing. The principal contribution lies in translating prediction confidence into intelligent computational decisions, enabling selective cloud utilisation and efficient resource allocation. This approach offers a scalable solution for orchard monitoring under limited connectivity and computational resources, with broader applications in precision agriculture and Internet of Things environments. Future research should explore learned decision policies, adaptive confidence thresholds and multimodal sensing incorporating RGB, thermal, humidity and soil data. Federated learning and fog computing could enhance distributed intelligence, while neural architecture search and Tiny Vision Transformers may improve embedded inference efficiency. Cloud-to-edge knowledge transfer and continuous model updating could further strengthen detection reliability under changing network conditions, disease prevalence and orchard environments.

FUNDING

This work was supported by the Science Committee of the Ministry of Science and Higher Education of the Republic of Kazakhstan (Grant No. AP23488699, "System for Detecting Phytopathogenic Infections in Domestic Fruit Crop Production Using Computer Vision and Artificial Olfaction").

CONFLICTS OF INTEREST

The authors declare no conflicts of interest.

References

[1] Savvidis, P., & Papakostas, G. A. (2021). *Remote crop sensing with iot and AI on the edge* In 2021 IEEE World AI IoT Congress (AIIoT), https://doi.org/10.1109/aiiot52608.2021.9454237

[2] da Silva, J. C. F., Silva, M. C., Luz, E. J. S., Delabrida, S., & Oliveira, R. A. R. (2023). Using mobile edge AI to detect and map diseases in citrus orchards. *Sensors*, *23*(4), 2165. https://doi.org/10.3390/s23042165

[3] Yuchao, X., Qian, W., Bingyuan, Z., Lingli, Z., Ni, R., & Meina, Z. (2025). Review on lightweight deep learning networks for object detection in crops. *Journal of Chinese Agricultural Mechanization*, *46*(3), 261. https://doi.org/10.13733/j.jcam.issn.2095-5553.2025.03.038

[4] Grujev, M., Stojanovic, D., Milosavljevic, A., Ilic, M., & Prodanovic, V. (2026). An efficient real-time apple disease classification approach using a novel lightweight neural network on an arduino edge device. *Internet of Things*, *35*, 101833. https://doi.org/10.1016/j.iot.2025.101833

[5] Zheng, Z., Zhang, Y., & Sun, L. (2025). Improved yolov8-based algorithm for citrus leaf disease detection. *IEEE Access*, *13*, 105888-105900. https://doi.org/10.1109/access.2025.3579074

[6] Zhou, S., Yin, W., He, Y., Kan, X., & Li, X. (2025). Detection of apple leaf gray spot disease based on improved yolov8 network. *Mathematics*, *13*(5), 840. https://doi.org/10.3390/math13050840

[7] Liu, B., Fan, H., Zhang, Y., Cai, J., & Cheng, H. (2024). Deep learning architectures for diagnosing the severity of apple frog-eye leaf spot disease in complex backgrounds. *Frontiers in Plant Science*, *14*. https://doi.org/10.3389/fpls.2023.1289497

[8] Wang, G., Sang, W., Xu, F., Gao, Y., Han, Y., & Liu, Q. (2025). An enhanced lightweight model for apple leaf disease detection in complex orchard environments. *Frontiers in Plant Science*, *16*. https://doi.org/10.3389/fpls.2025.1545875

[9] Zou, H., Lv, P., & Zhao, M. (2025). Detection of apple leaf diseases based on lightYOLO-AppleLeafDx. *Plants*, *14*(4), 599. https://doi.org/10.3390/plants14040599

[10] Liu, Y., Zhang, Y., Jiang, D., Zhang, Z., & Chang, Q. (2023). Quantitative assessment of apple mosaic disease severity based on hyperspectral images and chlorophyll content. *Remote Sensing*, *15*(8), 2202. https://doi.org/10.3390/rs15082202

[11] Karim, M. J., Goni, M. O. F., Nahiduzzaman, M., Ahsan, M., Haider, J., & Kowalski, M. (2024). Enhancing agriculture through real-time grape leaf disease classification via an edge device with a lightweight CNN architecture and Grad-CAM. *Scientific Reports*, *14*(1). https://doi.org/10.1038/s41598-024-66989-9

[12] Manoj, R. K., & Nandhini, S. A. (2025). Computer vision and AI-based apple plant disease detection framework in smart farming. In *Studies in Computational Intelligence* (pp. 269-286). Springer Nature Switzerland. https://doi.org/10.1007/978-3-032-05603-0_12

[13] Ramya, R. K., & Meenakshi, C. (2025). *An adaptive multi-stage ensemble learning approach*

for accurate diagnosis of tomato leaf diseases In 2025 6th International Conference on Inventive Research in Computing Applications (ICIRCA), https://doi.org/10.1109/icirca65293.2025.11089636

[14] Rehman, S., Attique Khan, M., Alhaisoni, M., Armghan, A., Alenezi, F., Alqahtani, A., Vesal, K., & Nam, Y. (2023). Fruit leaf diseases classification: A hierarchical deep learning framework. *Computers, Materials & Continua*, *75*(1), 1179-1194. https://doi.org/10.32604/cmc.2023.035324

[15] Kannan, E., M J, C. M. B., S, A. D., N, R. N., Begum, A., & D, H. (2024). Deep learning techniques advancements in apple leaf disease detection. *Procedia Computer Science*, *235*, 713-722. https://doi.org/10.1016/j.procs.2024.04.068

[16] Dhiman, P., Kaur, A., Hamid, Y., Alabdulkreem, E., Elmannai, H., & Ababneh, N. (2023). Smart disease detection system for citrus fruits using deep learning with edge computing. *Sustainability*, *15*(5), 4576. https://doi.org/10.3390/su15054576

[17] Dhiman, P., Manoharan, P., Lilhore, U. K., Alroobaea, R., Kaur, A., Iwendi, C., Alsafyani, M., Baqasah, A. M., & Raahemifar, K. (2023). PFDI: A precise fruit disease identification model based on context data fusion with faster-CNN in edge computing environment. *EURASIP Journal on Advances in Signal Processing*, *2023*(1), 1-18. https://doi.org/10.1186/s13634-023-01025-y

[18] Foroughi, A., Lloret, J., Jimenez, J. M., & Sendra, S. (2025). An edge computing wireless sensor network for diagnosing orange fruit disease. *Cluster Computing*, *28*(5). https://doi.org/10.1007/s10586-024-04999-y

[19] Thongnim, P., Sreekajon, J., & Pukseng, T. (2025). Optimizing fruit cultivation through an unsupervised learning approach with data scaling techniques for strategic insights. *Eurasian Journal of Mathematical and Computer Applications*, *13*(3), 112-127. https://doi.org/10.32523/2306-6172-2025-13-3-112-127

[20] Kang, P., & Somtham, A. (2022). An evaluation of modern accelerator-based edge devices for object detection applications. *Mathematics*, *10*(22), 4299. https://doi.org/10.3390/math10224299

[21] Galliera, R., & Suri, N. (2022). Object detection at the edge: Off-the-shelf deep learning capable devices and accelerators. *Procedia Computer Science*, *205*, 239-248. https://doi.org/10.1016/j.procs.2022.09.025

[22] Zagitov, A., Chebotareva, E., Toschev, A., & Magid, E. (2024). Comparative analysis of neural network models performance on low-power devices for a real-time object detection task. *Computer Optics*, *48*(2), 242-252. https://doi.org/10.18287/2412-6179-co-1343

[23] Tan, M., Pang, R., & Le, Q. V. (2020). *Efficientdet: scalable and efficient object detection* In 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), https://doi.org/10.1109/cvpr42600.2020.01079

[24] Fang, R., Gao, P., Zhou, A., Cai, Y., Liu, S., Dai, J., & Li, H. (2024). Feataug-detr: Enriching one-to-many matching for detrs with feature augmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *46*(9), 6402-6415. https://doi.org/10.1109/tpami.2024.3381961

[25] Jocher, G., Chaurasia, A., & Qiu, J. (2023). *Ultralytics YOLOv8*. In (Version 8.0.0) Ultralytics. https://ui.adsabs.harvard.edu/abs/2024zndo..10983461J/abstract