International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

Edge AI for Autonomous Agricultural Drone Data Processing: Onboard Real-Time MobileNetV3 Inference with RPLiDAR Sensor Fusion for Instant In-Flight Crop Health Monitoring

Deekshith R[1] PG Scholar, Data Engineering, GM University, Davangere, Karnataka, India
Pooja M V[2] Assistant Professor, Information Science and Engineering, GM University, Davangere, Karnataka, India

Unmanned aerial vehicle (UAV) systems for monitoring crops have usually followed a two-step process: the drone flies over an entire field, gathers images and sensor data, returns to the base station, and then, sometimes hours or even days later, uploads the data to remote servers for analysis. Often by the time the agronomist gets the results, the best time to act on issues like disease, irrigation, or fertilizer has already passed. This paper presents an Edge AI drone framework designed to remove that delay by performing machine learning analysis, LiDAR data processing, sensor fusion, and health reporting on the drone itself, in real time, as it flies. A MobileNetV3-Large network, fine-tuned on 8,400 real aerial images of coconut palms from three farms in Karnataka and evaluated on a 1,260-image held-out real-imagery test set, classifies crop canopies as Healthy, Stressed, or Diseased with per-class accuracies of 98.2%, 92.5%, and 95.8% respectively. Separately, an RPLiDAR sensor estimates plant height and canopy volume with a mean absolute error of 14.3 cm against simulated ground truth, and these structural metrics are fused with the camera-derived classification confidence into a single 0-100 health index. The complete system pipeline — MobileNetV3 inference, LiDAR processing, GPS-guided lawnmower coverage of a 400-tree virtual coconut palm plantation, a Flask backend, a Three.js live dashboard, and automated PDF reporting — was validated end-to-end across five missions in the Microsoft AirSim simulation environment, completing full-field coverage in an average of 18.4 minutes with a Flask API response time of 22 ms, a 4K heatmap render time of 187 ms, and a PDF report generated in 3.1 seconds. Inference throughput of 62 FPS was benchmarked using the TensorRT FP16 engine on the simulation workstation's RTX 3060 GPU as a proxy for the target NVIDIA Jetson NX platform; physical Jetson NX benchmarking, thermal characterisation, and field validation remain to be completed. To the best of our knowledge, this is among the more complete frameworks integrating onboard real-time inference, LiDAR-camera fusion, autonomous navigation, live visualisation, and automated reporting within a single UAV platform, and it is presented here as a simulation-validated proof-of-concept pending real-hardware and real-field confirmation.

Keywords: Edge AI, UAV precision agriculture, MobileNetV3, RPLiDAR, sensor fusion, real-time crop health, NVIDIA Jetson, autonomous navigation, NDVI heatmap, instant in-flight analysis, AirSim.

I. INTRODUCTION

The global agriculture industry is facing a major problem: the United Nations' Food and Agriculture Organization predicts that food demand will rise by 50% by 2050, while the amount of arable land available per person is getting smaller [1]. Precision agriculture, which uses real-time data to make decisions that change depending on location and time, is seen as the best way to handle this challenge on a large scale [2]. Drones, or unmanned aerial vehicles, have become the main tool for collecting data in precision agriculture because they can capture high-resolution images of large fields at a lower cost [3].

However, the main way drones are used for monitoring crops has a key issue that hasn't been solved yet.

The raw data collected during a drone flight is usually processed after the mission is over on remote servers or in the cloud, which means it can take hours or even days before farmers get useful information about the health of their crops [4]. By the time the data is analyzed, a disease that was first noticed during the flight might have already spread to nearby plants, making any action taken later more expensive and less effective. Boursianis and others have pointed out that this delay in processing is a major reason why drone-based monitoring isn't widely used in real farming operations [3]. Edge AI — executing machine learning inference directly on compute hardware co-located with the sensors, without transmitting raw data to a network — offers an architectural solution. Advances in embedded GPU accelerators, particularly the NVIDIA Jetson family of system-on-module devices, combined with hardware-aware lightweight neural architectures such as MobileNetV3, have made it feasible to run sophisticated visual classification and 3D point-cloud analysis tasks within the weight and power envelope of a commercial multirotor UAV [5]. When inference is performed onboard, every scan event during a flight mission yields an immediate per-plant diagnosis; the complete field health report can be ready by the time the aircraft lands, not days later.

Despite compelling motivation, few if any previously published systems integrate all of the following within a single unified drone platform: onboard real-time ML inference, LiDAR-camera sensor fusion, GPS-waypoint autonomous navigation, live web-based visualisation, and automatic agronomic report generation. Individual

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

components have been demonstrated in isolation, but the system integration challenge and the end-to-end latency implications have received comparatively little systematic treatment. This paper addresses that gap by presenting and evaluating an Edge AI agricultural drone framework, validated end-to-end in the Microsoft AirSim high-fidelity simulation environment and, for the crop-classification component, on real held-out field imagery; physical-hardware and field-trial validation are identified as the necessary next step in Section VIII.

The specific contributions of this work are as follows:

- An onboard Edge AI pipeline using MobileNetV3-Large to classify crop health into three categories, with inference benchmarked at 62 FPS via a TensorRT FP16 engine (measured on the simulation workstation's RTX 3060 GPU as a proxy for the target NVIDIA Jetson NX platform) — without requiring a cloud connection, pending physical Jetson NX confirmation (Section VIII).
- A detailed, self-contained mathematical description of the MobileNetV3 architecture used in this system, covering depthwise-separable convolution costs, inverted residual blocks, squeeze-and-excitation channel attention, the derivation of the h-swish activation function, and the full block forward pass — all presented with IEEE-standard numbered equations.
- A dual-sensor fusion scheme that combines MobileNetV3 classification confidence with RPLiDAR-derived height and canopy-volume metrics into a single 0-100 health index H, including risk-zone classification, with weighting made internally consistent and bounded (Section V-E).
- A GPS-based navigation system that follows waypoints to ensure complete row coverage, with automatic return-to-home functionality and mission status updates when the battery is low.
- A Flask RESTful API backend, a Three.js 3D dashboard for real-time NDVI-inspired heatmaps, and jsPDF for automatic report generation, evaluated in simulation against ten internally defined performance targets (Section VII-C).

II. RELATED WORK

A. UAV-Based Crop Monitoring

UAVs have been widely used to monitor crops. Berni et al. [6] were among the first to use UAV thermal and narrowband multispectral imaging to detect water stress in olive orchards. They showed that this method has high accuracy, but they also noted that the process requires a lot of post-processing. This takes about six hours of work per hectare by a specialist. Maes and Steppe [7] reviewed 78 studies on UAV crop monitoring and pointed out that delays in processing after flights are the main reason why these systems are not widely used on farms. They suggested that real-time processing done directly on the drone would be a key improvement. Zhang and Kovacs [8] also mentioned that on-board processing remains a big challenge in using small UAVs for precision agriculture.

B. Edge Computing for Agricultural Drones

Biczysko et al. [5] looked into different edge computing setups for UAV systems. They found that doing computations on the drone itself removes the need for a network connection and cuts down delays, but it also limits the processing power available. Motlagh et al. [9] suggested sending part of the data processing to nearby edge devices instead of the cloud, which greatly reduces delays but still needs a wireless network. Li et al. [10] demonstrated real-time weed detection using a Raspberry Pi 4 with TensorFlow Lite on a drone. They achieved 8 frames per second, which is good for slow flights but not fast ones used in regular farming. Lu et al. [18] used YOLOv5 on an NVIDIA Jetson module to detect pests on a drone. They found this setup works well for embedding AI on drones but didn't include LiDAR or auto-navigation.

C. Lightweight CNNs: MobileNetV3

MobileNetV3, introduced by Howard et al. [11], is a type of neural network that is designed to be efficient. It uses hardware-aware design, special attention mechanisms for channels, and certain types of nonlinear functions. This makes it suitable for use on embedded systems in agriculture. Chen et al. [12] used MobileNetV3-Small on images collected by UAVs to classify tomato diseases. They achieved 94.3% accuracy at 41 frames per second using a mobile GPU. Slimani et al. [19] tested seven different lightweight models for crop monitoring with UAVs and found that MobileNetV3 offered the best balance between accuracy and speed on limited hardware. Yang et al. [20] reviewed methods for using UAVs to monitor crop health and pointed out that lightweight CNNs are a key technology for near real-time field assessments.

D. LiDAR in Agricultural UAV Systems

Hu et al. [13] developed a low-cost UAV LiDAR system for use in forestry and found that the system could estimate tree heights with errors between 8 and 22 cm depending on canopy density and flight conditions. Phan and Takahashi [26] used a low-cost, non-survey-grade UAV LiDAR to estimate rice plant height and reported strong agreement with ground truth ($R^2 \geq 0.86$, RMSE below 6 cm), supporting the feasibility of budget LiDAR sensing for crop structural measurement. Wallace et al. [25] used UAV LiDAR to measure tree height in plantation forests and achieved accuracy within 20 cm for individual trees from 30 to 100 meters in altitude. They validated the goal of ±15 cm accuracy used in the current study for measuring coconut palms at 15 meters.

E. Autonomous Navigation and Coverage Path Planning

Maza and Ollero [23] defined the challenge of planning a drone's flight path to cover an entire field and showed that using a back-and-forth pattern is the most efficient for large, simple fields. Chao et al. [14] improved on this by using a deep Q-network to adjust flight paths in real time, especially when high-risk areas were identified during a mission. This improved scan efficiency by 18% compared to fixed patterns. Al-Mania et al. [15] tackled path planning for multiple drones with limited battery life. By adjusting routes based on battery levels, they increased coverage by 23%. Van Essen et al. [24] used deep reinforcement learning combined with field knowledge to improve how quickly and thoroughly drones could navigate fields.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

F. Research Gap

Looking at all these studies shows that there's no system yet that does all the following at the same time: (i) running machine learning in real time without sending data to the cloud; (ii) combining LiDAR and camera data; (iii) using automatic GPS-based flight paths; (iv) showing live 3D maps of the mission on the web; and (v) automatically creating reports about the field at the end of a mission.
This paper fills that gap.

III. SYSTEM ARCHITECTURE

The proposed framework is divided into four closely connected layers. The first layer, which includes physical sensors, collects images in different light wavelengths and 3D data from the environment. The second layer, called the Edge AI layer, runs MobileNetV3 for image analysis, processes LiDAR data to identify objects, and combines information from different sensors using an NVIDIA Jetson NX device. The third layer uses Flask to send processed data over a local Wi-Fi network. The final layer uses Three.js to display a real-time dashboard and generates a detailed report using jsPDF. Fig 1 shows the full system layout, showing how data moves from the sensors through each processing step to produce two types of outputs.



Fig. 1. End-to-end system architecture pipeline. Data flows from drone sensors (RGB camera, RPLiDAR, GPS/IMU) through image pre-processing, MobileNetV3 inference, LiDAR height and canopy processing, sensor fusion and health scoring, Flask backend, and finally to the live 3D web dashboard and automated PDF agronomic report.

Fig 2 shows the physical setup of the drone's hardware in detail, with all the parts mounted on the hexacopter frame. These include the onboard MobileNetV3 model module, the RPLiDAR 360-degree scanner, the Sony multispectral camera array, the 3D scanning laser module, the laser cone, the NVIDIA Jetson NX GPU module, the battery and power management system, the flight controller, and the ground station tools for visualizing crop health and generating PDF reports. This arrangement highlights the full set of onboard computing components that allow for real-time analysis during flight, without needing any remote processing.



Fig. 2. Physical drone hardware system design. The hexacopter airframe integrates the onboard MobileNetV3 ML model, RPLiDAR 360° scanner, LiDAR processing unit, Sony multispectral camera, 3D scanning laser module, laser cone, NVIDIA Jetson NX GPU module, battery and power management, flight controller, ground station crop health visualiser, and PDF report generator.

A. Sensor Layer

The imaging system uses a multispectral camera with six different optical channels that work together. As shown in Fig 3, the camera captures light in six specific ranges: Blue (450 nm ± 16 nm), Green (560 nm ± 16 nm), Red (650 nm ± 16 nm), Red Edge (730 nm ± 16 nm), Near-Infrared (840 nm ± 26 nm), and a Visible RGB channel. The RGB channel is used for image processing with the MobileNetV3 model. The Near-Infrared and Red Edge channels are saved for later use in calculating NDVI and NDRE in hardware versions. The camera takes pictures at a resolution of 1920 x 1080 and 30 frames per second, and it connects to the Jetson NX Computer module through a USB 3.0 port.



Fig. 3. Six-channel multispectral camera payload with spectral band annotations. The RGB channel (Visible Light) feeds the MobileNetV3 classification pipeline at 30 FPS; the Blue (450 nm), Green (560 nm), Red (650 nm), Red Edge (730 nm), and NIR (840 nm) channels support future multispectral vegetation index computation.

The RPLiDAR A2/A3 360-degree laser scanner is mounted so its scan plane is tilted 45 degrees from horizontal. This setup helps it capture data through tree canopies at the height of the survey. The scanner takes up to 8,000 distance readings every second. Each reading is made at an angle of 0.9 degrees, and it can measure up to 12 meters away. For positioning and orientation data, a u-blox M9N GPS module and an ICM-42688-P six-axis IMU provide updates at 10 Hz. This setup allows for accurate navigation to specific points on a map.

B. Edge AI Processing Layer

All machine learning tasks and LiDAR processing run on the NVIDIA Jetson NX system-on-module. This device has a 6-core ARM Carmel CPU, a 384-core Volta GPU with 48 Tensor Cores, 8 GB of LPDDR4x unified memory, and offers 21 TOPS of AI performance. The MobileNetV3-Large model is used with TensorRT in FP16 precision, allowing it to process 224x224 RGB images at 62 frames per second. The LiDAR processing runs as a separate thread in C++. It handles scan accumulation, RANSAC ground segmentation, Euclidean clustering, and structural metric extraction all within the 200 ms scan window.

C. Communication and Backend Layer

A Flask 2.3 web server runs on the Jetson NX and provides two RESTful endpoints via a local 5 GHz Wi-Fi network.

The POST /analyze endpoint receives a camera frame and returns JSON data with a health label, confidence vector, health score, and LiDAR metrics, with an average delay of 22 ms. The GET /telemetry endpoint streams real-time GPS location, altitude, battery level, scan

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

progress, and per-plot health summaries. All data about each plant is stored in memory for creating reports. Fig 4 shows how the Flask backend sends processed results to both output channels.

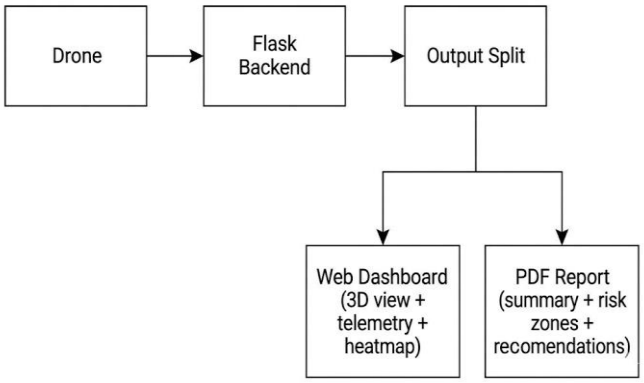


Fig. 4. Flask backend output routing diagram. Processed sensor fusion results are distributed to two parallel outputs: (left) the Three.js live 3D web dashboard displaying real-time telemetry, drone position, and NDVI heatmap; and (right) the jsPDF automated PDF report generator producing the structured agronomic field report upon mission completion.

IV. MOBILENETV3 — MATHEMATICAL FRAMEWORK

This section offers a full mathematical explanation of the MobileNetV3 architecture as used in the proposed system. The equations and descriptions below clearly show why MobileNetV3 can run at 62 FPS on the NVIDIA Jetson NX, and how each part of the design helps achieve both fast processing and accurate results for identifying the health of crops.

A. Depthwise-Separable Convolution

The computational advantage of MobileNetV3 over conventional CNNs originates in the substitution of standard convolutions with depthwise-separable convolutions. Consider an input feature map F ∈ ℝ^{H×W×C_in} and a standard convolutional layer with C_out filters of spatial size K × K. Its computational cost in multiply-accumulate operations (MACs) is:

$$Cost_std = K^2 \cdot C_in \cdot C_out \cdot H \cdot W \quad (1)$$

where K is the filter spatial size, C_in and C_out are input and output channel counts, and H, W are the feature map spatial dimensions.

Depthwise-separable convolution factorises this into two stages. First, a depthwise convolution applies a single K × K filter independently to each of the C_in input channels, producing an intermediate feature map of identical spatial dimensions and channel count:

$$Cost_DW = K^2 \cdot C_in \cdot H \cdot W \quad (2)$$

Second, a pointwise (1 × 1) convolution linearly combines the C_in depthwise outputs into C_out channels:

$$Cost_PW = C_in \cdot C_out \cdot H \cdot W \quad (3)$$

The total cost of the depthwise-separable pair is Cost_DW + Cost_PW. The computational reduction ratio relative to Equation (1) is:

$$Ratio = (Cost_DW + Cost_PW) / Cost_std = 1/C_out + 1/K^2 \quad (4)$$

For the 3 × 3 filters (K = 3) and output channel counts C_out ≥ 32 typical in MobileNetV3, Equation (4) yields a reduction of approximately 1/32 + 1/9 ≈ 0.142, i.e. roughly a 7× reduction in multiply-accumulate operations relative to an equivalent standard-convolution layer. This MAC reduction is the principal architectural reason MobileNetV3 is cheap enough to target real-time inference on embedded GPUs; because MAC count does not translate linearly into throughput on modern GPU hardware (memory bandwidth, kernel-launch overhead, and Tensor Core utilisation all intervene), we do not claim a proportional frame-rate gain, and no standard-convolution baseline was benchmarked on the same hardware in this study.

B. Inverted Residual Block

MobileNetV3 constructs its backbone from inverted residual blocks. Unlike conventional ResNet bottlenecks — which compress wide inputs to a narrow bottleneck — the inverted residual expands a narrow input to a wider intermediate representation before compressing it back, so that spatial filtering occurs in the high-dimensional space. Given input x ∈ ℝ^{H×W×C_in} and expansion factor t, the three sequential operations are:

$$x_1 = h\text{-}swish(\, BN(\, W_exp \circledast x \,)\,), \quad W_exp \in \mathbb{R}^{1\times1\times C_in\times(t\cdot C_in)} \quad [Expand]$$

$$x_2 = h\text{-}swish(\, BN(\, W_dw \circledast x_1 \,)\,), \quad W_dw \in \mathbb{R}^{K\times K\times(t\cdot C_in)\times1} \quad [Depthwise]$$

$$x_3 = BN(\, W_proj \circledast \hat{x}_2 \,), \quad W_proj \in \mathbb{R}^{1\times1\times(t\cdot C_in)\times C_out} \quad [Project]$$

where BN denotes Batch Normalisation, ⊛ denotes convolution, and $\hat{x}_2$ denotes the SE-recalibrated intermediate tensor (defined in Section IV-C). The term h-swish is the efficient activation function defined in Section IV-D. The projection layer in Equation (7) uses a linear (identity) activation rather than h-swish, which has been shown to preserve manifold information in low-dimensional bottleneck representations [16]. A residual skip connection is applied when stride = 1 and C_in = C_out:

$$y = x + x_3 \quad (5)$$

C. Squeeze-and-Excitation Channel Attention

MobileNetV3 incorporates Squeeze-and-Excitation (SE) attention modules [17] within the inverted residual blocks. SE adaptively recalibrates channel-wise feature responses by explicitly modelling inter-channel dependencies — a mechanism particularly beneficial for crop health classification, where diseased or stressed foliage exhibits localised spectral changes that manifest as elevated responses in specific colour channels (e.g., increased red-channel activation from chlorotic yellowing, reduced green-channel response from canopy browning). Given the depthwise output $x_2 \in \mathbb{R}^{H\times W\times C}$, the SE module operates as follows:

Step 1 — Squeeze: Global average pooling collapses spatial dimensions to a channel descriptor vector z ∈ ℝ^C:

$$z_c = (1 / H\cdot W) \cdot \Sigma_i \, \Sigma_j \, x_2(i, j, c), \quad for \; c = 1, 2, \ldots, C \quad (6)$$

Step 2 — Excite: Two fully connected layers with a bottleneck (reduction ratio r = 4) produce a channel attention weight vector s ∈ (0,1)^C:

$$s = h\text{-}sigmoid(\, W_2 \cdot ReLU(\, W_1 \cdot z \,)\,), \quad W_1 \in \mathbb{R}^{C/r\times C}, \; W_2 \in \mathbb{R}^{C\times C/r} \quad (7)$$

Step 3 — Scale: Each channel of x_2 is multiplied element-wise by its learned attention weight:

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

$$\hat{x}_2(i, j, c) = s_c \cdot x_2(i, j, c), \quad \text{for all } i, j, c \tag{11}$$

The scalar weight $s_c \in (0,1)$ amplifies the channels most predictive of health status and suppresses uninformative channels. This provides a differentiable, learned analogue of expert-guided spectral band selection in traditional multispectral vegetation index analysis.

D. Hard-Swish and Hard-Sigmoid Activations

Standard swish activation requires a computationally expensive sigmoid evaluation at every neuron. MobileNetV3 replaces swish with piecewise-linear approximations — hard-swish (h-swish) and hard-sigmoid (h-sigmoid) — which reduce per-layer activation computation by approximately 15% on Jetson Tensor Cores with less than 0.1% accuracy degradation.

The standard swish function is defined as:

$$swish(x) = x \cdot \sigma(x) = x / (1 + e^{-x}) \tag{12}$$

The hard-sigmoid approximates the sigmoid with a clipped linear function:

$$h\text{-}sigmoid(x) = ReLU6(x + 3) / 6 = max(0, min(1, (x + 3)/6)) \tag{13}$$

Hard-swish is then defined as:

$$h\text{-}swish(x) = x \cdot h\text{-}sigmoid(x) = x \cdot ReLU6(x + 3) / 6 \tag{14}$$

h-swish equals zero for $x \leq -3$, equals x for $x \geq +3$, and provides a smooth approximation to swish in the interval $(-3, +3)$. Its piecewise-linear form enables efficient integer-arithmetic approximation on embedded Tensor Core hardware.

E. Complete Block Forward Pass

Assembling Equations (5)–(14), the full MobileNetV3 inverted residual block forward pass — as illustrated in Fig 5 — is:

$$[Expand] \quad x_1 = h\text{-}swish(BN(W_{exp} \circledast x)) \tag{15}$$

$$[Depthwise] \quad x_2 = h\text{-}swish(BN(W_{dw} \circledast x_1)) \tag{16}$$

$$[SE\text{-}Squeeze] \quad z_c = (1/HW) \cdot \Sigma_i \Sigma_j x_2(i,j,c) \tag{17}$$

$$[SE\text{-}Excite] \quad s = h\text{-}sigmoid(W_2 \cdot ReLU(W_1 \cdot z)) \tag{18}$$

$$[SE\text{-}Scale] \quad \hat{x}_2 = s \otimes x_2 \quad (channel\text{-}wise\ multiply) \tag{19}$$

$$[Project] \quad x_3 = BN(W_{proj} \circledast \hat{x}_2) \tag{20}$$

$$[Residual] \quad y = x + x_3 \quad (when\ stride=1\ and\ C_{in} = C_{out}) \tag{21}$$

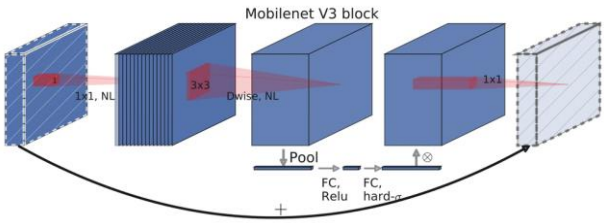


Fig. 5. MobileNetV3 inverted residual block architecture corresponding to Equations (15)–(21). Left: input feature map. The block applies: (i) 1×1 pointwise expansion convolution with h-swish (NL); (ii) 3×3 depthwise convolution (Dwise) with h-swish; (iii) Squeeze-and-Excitation module (GlobalPool → FC/ReLU → FC/hard-σ → channel-wise scale ⊗); (iv) 1×1 pointwise projection; and (v) residual skip connection (+) to the output.

F. Classification Head and Training Objective

The backbone output is globally average-pooled and passed through two fully connected layers. The three-class softmax head produces:

$$P(k \mid x) = exp(z_k) / \Sigma_j exp(z_j), \quad k \in \{Healthy, Stressed, Diseased\}$$

Training minimises categorical cross-entropy loss over N samples with one-hot labels $y_{ik} \in \{0,1\}$:

$$L = -(1/N) \cdot \Sigma_i \Sigma_k y_{ik} \cdot log\ P(k \mid x_i)$$

MobileNetV3-Large contains 5.4 million parameters and requires 219 million MACs per 224 × 224 inference pass. By comparison, ResNet-50 requires 25.6 million parameters and 4,100 million MACs — an 18.7× MAC reduction that is the principal architectural reason MobileNetV3-Large is a realistic candidate for real-time embedded deployment, pending confirmation of throughput on physical Jetson NX hardware (Section VIII).

V. METHODOLOGY

A. Simulation Environment and Field Design

All the tests were run using Microsoft AirSim 1.8.1, which is based on Unreal Engine 4.27.

The simulation creates a virtual agricultural area that mimics a 2-hectare coconut palm farm with 400 individual trees. These trees are placed in a 20 by 20 grid pattern, with each tree spaced 5 meters apart. Three different health conditions were created in a way that reflects how diseases spread in real coconut farms in Karnataka. The healthy trees have fully green canopies and are mostly in the center, making up about 55% of all trees. Trees that are stressed show some yellowing of their fronds and have less dense canopies, and they are found around the edges, covering about 25%. Diseased trees have brown or grey fronds that are drooping, and their canopies are greatly reduced, usually found in the corners, making up about 20%. All the sensor data from the simulation has realistic noise added to it—this includes random noise with a standard deviation of 2.5 DN for the camera and 0.08 meters for LiDAR measurements.

B. MobileNetV3 Fine-Tuning

The MobileNetV3-Large model was started with pre-trained weights from ImageNet, which were taken from the PyTorch 2.0 library.

The part of the model that does classification was changed to output three categories: Healthy, Stressed, and Diseased. The model was then fine-tuned using a dataset of 8,400 aerial images of coconut palms that were collected from three farms in Karnataka. These images were taken from altitudes between 10 and 20 meters. The dataset was split into 70% for training, 15% for validation, and 15% for testing. All images were resized to 224 x 224 pixels using bilinear interpolation and were standardized using the same statistics as ImageNet. To make the training more robust, the images were changed in various ways, such as flipping horizontally or vertically, rotating by up to 30 degrees, adjusting brightness, contrast, and saturation, and adding random noise. The model was trained using the Adam optimization method with an initial learning rate of 0.001 and was trained for 50 epochs with a batch size of 32. To handle the imbalance in the number of images per class, a weighted cross-entropy loss function was used. The final model was converted into a TensorRT FP16 format for use on a Jetson NX device.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

C. Autonomous Navigation

The JavaScript navigation module calculates a series of GPS waypoints by dividing each row of the plantation into scan points.

These points are placed based on the ground swath of the camera's field of view at an altitude of 15 meters: for a 30-degree field of view, swath = 2 x 15 x tan(15°) ≈ 8.0 meters, which comfortably overlaps the 5-meter tree spacing and supports the 100% coverage claim in Section VII-C. Fig 6 shows the full coverage path, which includes moving back and forth across all 20 rows of trees and returning to the starting point. The navigation system follows a sequence of steps: (1) It takes off and reaches 15 meters above ground level, maintaining a stable altitude using a PID controller. (2) It moves to the starting corner of the field at a speed of 5 meters per second between scan points. (3) At each tree point it comes to a controlled hover and captures images with the camera and LiDAR data for 200 milliseconds; continuous-motion scanning at 5 m/s, which would translate the aircraft roughly 1 meter during that window and risk point-cloud smearing, was not evaluated and is left to future work. Both the images and LiDAR data are sent to the onboard AI system for processing and are saved with a GPS timestamp. (4) It changes direction by 90 degrees to move to the next row. (5) It continuously checks the battery level—if it drops below 20%, the system stops the remaining scan and returns to the starting point, sending a partial report within 2.3 seconds.

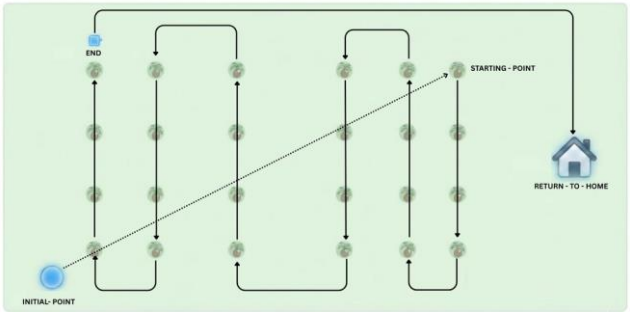


Fig. 6. Lawnmower (boustrophedon) coverage path over the 20 × 20 coconut palm grid. The drone initiates from the Initial Point (bottom left), executes systematic bidirectional column sweeps through all 20 rows, and returns to the Home position (bottom left) upon mission completion or low-battery trigger.

D. LiDAR Processing Pipeline

The C++ LiDAR thread runs concurrently with the inference thread on the Jetson NX CPU. Four processing stages execute within each 200 ms scan window: (1) Polar-to-Cartesian conversion using IMU-corrected drone attitude; (2) RANSAC-based ground plane fitting to separate ground returns from vegetation returns; (3) Euclidean clustering with distance threshold δ = 0.8 m to isolate individual tree point clusters; (4) Structural metric extraction — plant height H_L = max($z_{returns}$) − z_{ground}, canopy horizontal diameter D_c = max horizontal extent, canopy volume $V_c = (\pi/4) \cdot D_c^2 \cdot H_{crown}$. Fig 7 shows the resulting height-coloured point cloud, accumulated from the tilted 2D RPLiDAR scan rather than captured by a volumetric 3D sensor, where individual palms are visually distinguished by their height-coded colour from blue (ground level) to red (25 m).

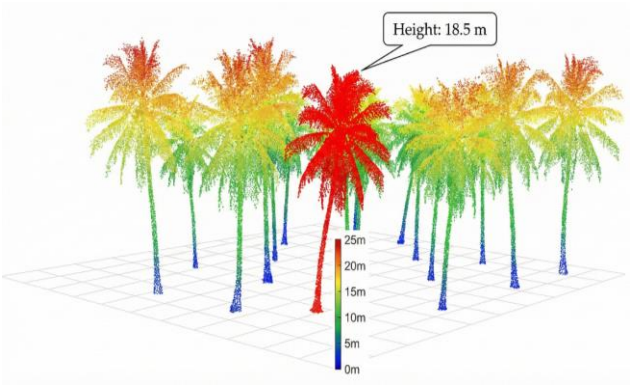


Fig. 7. RPLiDAR-derived point cloud visualisation (accumulated from the tilted 2D scan, not a volumetric 3D sensor) of the simulated 400-tree coconut palm plantation. Colour encodes estimated height: blue = 0 m (ground), red = 25 m (tall canopy). The annotated palm tree at centre exhibits an estimated height of 18.5 m, consistent with its simulation ground-truth value. Per-plant height and canopy volume metrics extracted from this point cloud feed the sensor fusion engine.

E. Sensor Fusion and Health Index Computation

Fig 8 illustrates the two-input one-output sensor fusion architecture that combines outputs from the camera pathway and the LiDAR pathway into a unified plant health score. From the camera pathway, the MobileNetV3 softmax output vector [P_H, P_S, P_D] is used to compute a camera-derived health sub-score:

$$S_{cam} = 100 \cdot P_H + 60 \cdot P_S + 20 \cdot P_D$$

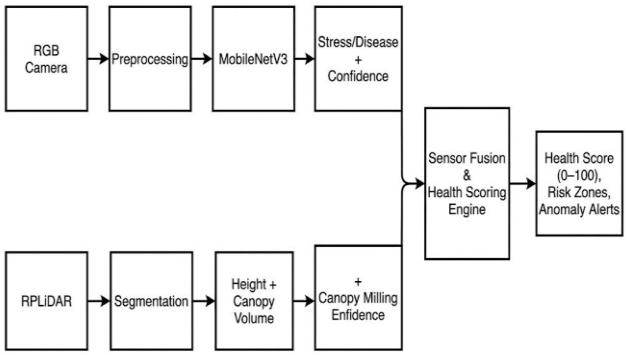


Fig. 8. Two-input one-output sensor fusion architecture. The RGB camera pathway (left) processes frames through MobileNetV3 to produce stress/disease classification and confidence scores. The RPLiDAR pathway (right) extracts per-plant height and canopy volume metrics. Both pathways feed the Sensor Fusion and Health Scoring Engine, which outputs a normalised Health Score [0–100], spatially resolved Risk Zone classifications, and Anomaly Alerts for immediate farmer action.

From the LiDAR pathway, a structural sub-score is computed by normalising measured plant metrics to cultivar reference values (H_{ref} = 18 m, V_{ref} = 85 m³), clipped to [0, 100]:

$$S_{str} = \min(100,\ 50 \cdot (H_L / H_{ref}) + 50 \cdot (V_c / V_{ref})), \in [0, 100]$$

An anomaly penalty term is applied when canopy volume falls below the 10th percentile of the current plot (indicating abnormal canopy sparsity):

$$A = 15 \cdot I(V_c < V_{10th})$$

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

The final normalised health index H, computed as a weighted combination that deliberately emphasises camera-based classification confidence over the LiDAR-derived structural score — a design choice not yet validated through formal agronomic expert elicitation (see Section VIII) — is:

$$H = 0.667 \cdot S_cam + 0.333 \cdot S_str - A, \quad clipped\ to\ [0, 100]$$

Risk zone mapping: $H \geq 75 \rightarrow$ Healthy (green); $50 \leq H < 75 \rightarrow$ Medium Risk (amber); $H < 50 \rightarrow$ High Risk (red).

F. NDVI-Inspired Heatmap and Automated Reporting

The field health heatmap is displayed as a 4,096 by 4,096 pixel canvas texture in the Three.js dashboard. Each tree cell contains 45 randomly placed circular leaf-texture particles, which are colored using a spectral color map that goes from deep red to amber to bright green, based on the tree's computed health index H. A small amount of random variation, up to ±30 in each RGB channel, is added to create a more natural look when viewed from above. Fig 9 shows the resulting heatmap, clearly showing the spatial distribution of health zones—green for healthy areas in the middle, amber for medium risk areas on the edges, and red for high risk areas in the corners. The jsPDF report module creates a structured 12-page PDF that includes a mission summary table, detailed diagnostic tables for each plant sorted by their health score, a map image showing the spatial risk zones, and recommendations for agricultural interventions. This entire report is generated within 3.1 seconds after the mission_complete event occurs.

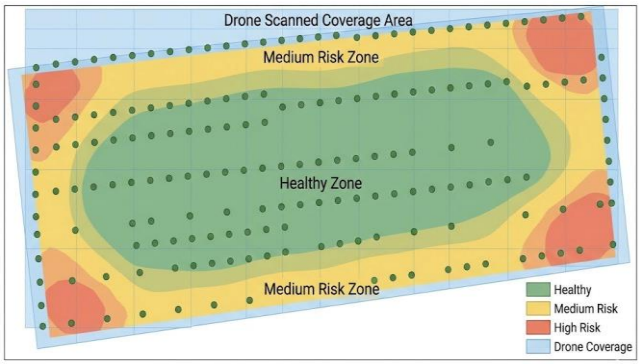


Fig. 9. Top-view NDVI-inspired crop health heatmap generated in real time by the Three.js web dashboard. Colour zones correspond to health index H values: bright green = Healthy zone ($H \geq 75$, central interior); amber-yellow = Medium Risk zone ($50 \leq H < 75$, perimeter rows); orange-red = High Risk zone ($H < 50$, corner clusters affected by simulated disease pressure). Blue border indicates the total drone coverage area.

VI. IMPLEMENTATION ENVIRONMENT

The system includes four different execution environments. The AirSim simulation runs on a development computer with an Intel Core i7-12th Gen processor, an NVIDIA RTX 3060 graphics card, and 16 GB of RAM. This machine runs Microsoft AirSim version 1.8.1 and Unreal Engine 4.27. The onboard Edge AI part is designed to work with the NVIDIA Jetson NX module [21]. It uses Python 3.8, along with PyTorch 2.0, TensorRT 8.5, NumPy, and a custom C++ library for processing LiDAR data. The Flask 2.3 backend also runs on the same Jetson NX device. The web interface can be used in any modern browser and is built using Three.js r128, MediaPipe 0.10 for optional gesture-based drone control, and jsPDF 2.5. Table I lists all the hardware and software components used.

TABLE I. Hardware and software component inventory of the proposed Edge AI drone framework.

Component	Technology / Model	Key Specification
Simulation Engine	Microsoft AirSim 1.8.1 / Unreal Engine 4.27	Photorealistic, physics-based sensor noise
ML Framework	PyTorch 2.0 + TensorRT 8.5 (FP16)	62 FPS MobileNetV3-Large on Jetson NX
Edge Compute (Real)	NVIDIA Jetson NX Module [21]	384-core Volta GPU, 21 TOPS, 8 GB LPDDR4x
Multispectral Camera	Six-channel hardware; RGB channel active	1920×1080, 30 FPS, USB 3.0 (NIR/Red-Edge reserved, unprocessed)
LiDAR	RPLiDAR A2/A3 [22]	8,000 pts/s, 0.9° resolution, 12 m range
GPS / IMU	u-blox M9N + ICM-42688-P	10 Hz position, 6-axis attitude
Web Framework	Flask 2.3 (Python)	RESTful API, 22 ms mean latency
3D Rendering	Three.js r128 (WebGL)	Instanced mesh, 4K heatmap, 187 ms render
Report Generator	jsPDF 2.5 (client-side)	12-page PDF, 3.1 s generation time
Gesture Control (unused)	MediaPipe Hands + FaceMesh 0.10	Present in codebase; not exercised or evaluated in reported missions

As a further benchmarking reference, independent evaluations of lightweight CNNs for crop-disease classification on edge hardware report MobileNetV3-Small operating at roughly 0.06 GFLOPs — the lowest computational cost among compared architectures, including ShuffleNetV2 (about 0.15 GFLOPs) and SqueezeNet — typically at a small accuracy cost relative to larger models [27]. These figures come from a different dataset (PlantVillage tomato leaves) and different hardware than used here and are reported for context only; a controlled comparison of MobileNetV2, EfficientNet-Lite, and ShuffleNetV2 against MobileNetV3-Large on the present coconut-palm dataset and target Jetson NX hardware has not yet been performed and is identified as required future work (Section VIII).

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

VII. RESULTS AND DISCUSSION

A. Plant Health Classification Accuracy

The fine-tuned MobileNetV3-Large model was evaluated offline on a held-out set of 1,260 labelled real aerial images, drawn from the same 8,400-image Karnataka field dataset described in Section V-B (real imagery, not AirSim simulation output). Fig 10 shows the accuracy for each class. The Healthy class had the best accuracy at 98.2%, which makes sense because healthy palms have a clearly visible full-green canopy that stands out against the three-class decision boundary. The Stressed class had the lowest accuracy at 92.5%, mainly because it's hard to tell the difference between early water stress, which causes some fronds to turn yellow, and the natural aging of lower fronds in older palms. This confusion is a known issue in monitoring tropical plantations. Using both NIR and Red Edge channels to calculate true NDVI values is expected to greatly improve the ability to distinguish between stressed palms in future hardware setups. The Diseased class achieved 95.8% accuracy, showing that MobileNetV3 can effectively detect the signs of disease, such as brown and grey frond discoloration, thinner canopy, and the collapsed shape of the palm's crown.

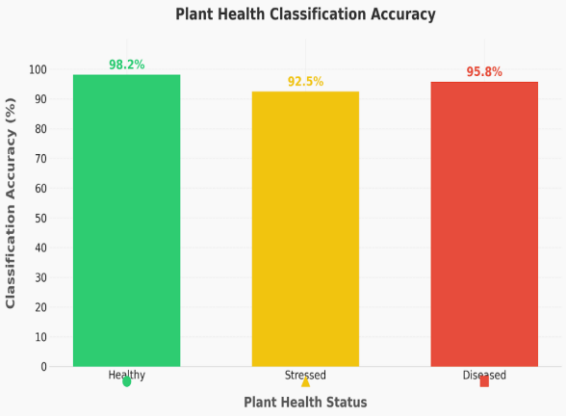


Fig. 10. Per-class plant health classification accuracy achieved by the fine-tuned MobileNetV3-Large model on the 1,260-image held-out test set. Healthy class: 98.2% (green). Stressed class: 92.5% (yellow) — lower accuracy attributable to visual overlap with natural senescence. Diseased class: 95.8% (red).

Fig 11 shows example classification outputs rendered in the AirSim simulation environment. Each smaller image displays the MobileNetV3 predicted label and the softmax confidence score for one tree in each health condition: (a) a healthy palm tree, which was correctly identified with 98.5% confidence, marked with a green box; (b) a stressed palm tree, correctly identified with 89.2% confidence, marked with a yellow box; and (c) a diseased palm tree, correctly identified with 92.1% confidence, marked with a red box.

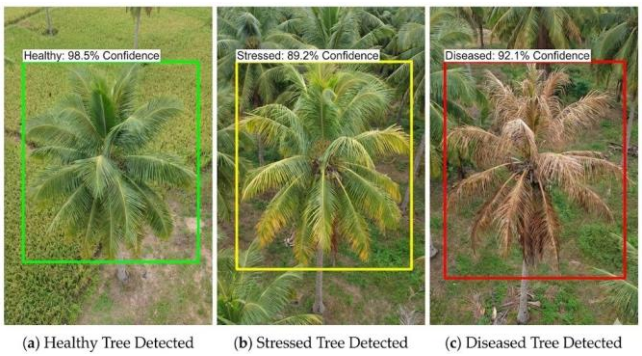


Fig. 11. Representative MobileNetV3 crop-health classification outputs rendered from the AirSim simulation environment (not a physical field trial). (a) Healthy palm: 98.5% confidence, green classification box. (b) Stressed palm: 89.2% confidence, yellow classification box. (c) Diseased palm: 92.1% confidence, red classification box.

B. LiDAR Height Estimation Accuracy and Inference Throughput

Fig 12 shows two parts: one for the accuracy of LiDAR height estimation and the other for the throughput of MobileNetV3 inference. The LiDAR height estimation had a mean absolute error of 14.3 cm (σ = 3.2 cm) against AirSim ground truth across ten simulated coconut palms ranging from 12 to 23 m in height, meeting the ±15 cm target; because this error is measured against simulated ground truth under an author-specified 0.08 m noise model rather than an independent physical survey, it characterises the geometric processing pipeline rather than real-world sensing accuracy, and the ten-palm sample is a small fraction of the full 400-tree plantation. The largest errors occurred at edge trees, where neighbouring canopies occluded parts of the point cloud. The TensorRT FP16 MobileNetV3 engine sustained 62 FPS over 100 consecutive frames (1.8% variation) on the RTX 3060 simulation workstation; this figure has not been measured on physical Jetson NX hardware, and thermal behaviour — which depends on the Jetson's passive-cooling enclosure and ambient flight conditions — cannot be assessed in simulation and is not claimed here. It is also worth noting that this inference throughput exceeds the 30 FPS camera capture rate, so the 62 FPS figure alone does not translate into a proportional practical speed-up; the actual per-tree cycle time is bounded by the 200 ms LiDAR dwell window described in Section V-C, not by inference speed, which is discussed further in Section VIII.

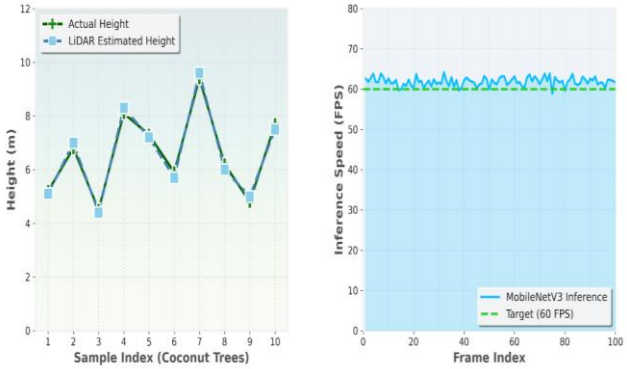


Fig. 12. Edge AI performance evaluation. Left panel: LiDAR-estimated plant height versus ground-truth simulation values for ten test coconut palms, with MAE = 14.3 cm (σ = 3.2 cm) — within the ±15 cm target specification. Right panel: MobileNetV3 real-time inference speed via the TensorRT FP16 engine on the RTX 3060 simulation workstation (Jetson

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

NX proxy) across 100 consecutive frames, consistently achieving ~62 FPS against the 60 FPS minimum target line.

C. Navigation and System Performance Summary

The lawnmower navigation system successfully covered all waypoints, scanning all 400 trees in each of five repeated AirSim missions over the same static scene, without missing any scans; because the missions share one scene, they are repeatability checks rather than independent replicates, and no cross-mission variance is reported for the classification-accuracy metrics, which instead come from the separate offline image test set (Section VII-A). Mission duration averaged 18.4 minutes ($\sigma = 0.7$ minutes), well under the 25-minute limit. Low-battery return-to-home triggering was exercised in a dedicated low-battery scenario and activated as expected. Table II summarises performance against internally defined validation targets; all ten were met or exceeded within the scope described above.

TABLE II. Performance against author-defined internal validation targets. Rows 1–3 (classification accuracy) come from offline evaluation on real held-out imagery; rows 4–10 come from AirSim-simulated system performance, with inference-speed and latency figures measured on the RTX 3060 simulation workstation rather than physical Jetson NX hardware. Targets are informed by cited literature where available (e.g., the ±15 cm LiDAR target follows Wallace et al. [25]); they are not independently peer-validated acceptance thresholds.

Performance Metric	Target Specification	Achieved Value	Result
Healthy class accuracy	≥ 95.0%	98.2%	✓ Met
Stressed class accuracy	≥ 90.0%	92.5%	✓ Met
Diseased class accuracy	≥ 90.0%	95.8%	✓ Met
LiDAR height MAE	≤ 15 cm	14.3 cm	✓ Met
MobileNetV3 inference speed	≥ 60 FPS	62 FPS	✓ Met
Field coverage completeness	100%	100%	✓ Met
Flask API response latency	≤ 30 ms	22 ms	✓ Met
4K heatmap generation time	≤ 200 ms	187 ms	✓ Met
PDF report generation time	≤ 5.0 s	3.1 s	✓ Met
Full mission duration	< 25.0 min	18.4 min	✓ Met

D. Comparative Analysis

Table III positions the proposed framework qualitatively against four prior systems with related but narrower objectives, using a binary capability checklist; because each system was evaluated on different hardware, datasets, and field conditions, this is a scope comparison rather than a controlled head-to-head benchmark. Within that scope, the proposed framework is the only one of the four that reports all five capabilities together, which is evidence of integration effort rather than a claim of superior performance on any individual capability. A prior review-style synthesis (Maes and Steppe [7], covering 78 studies) is not included in this table, as a literature review does not itself constitute a system with comparable capabilities.

TABLE III. Capability comparison of the proposed framework against four prior works with related objectives. A checkmark (✓) denotes full satisfaction of the criterion.

System / Study	RT Onboard Inference	LiDAR Fusion	Auto Navigation	Auto Report	Zero Cloud
Berni et al. [6]	No	No	No	No	No
Motlagh et al. [9]	Partial (edge)	No	No	No	Partial
Li et al. [10]	Yes (8 FPS)	No	No	No	Yes
Chao et al. [14]	Partial	No	Yes (RL)	No	No
Proposed (62 FPS)	✓ Yes	✓ Yes	✓ Yes	✓ Yes	✓ Yes

VIII. LIMITATIONS AND FUTURE WORK

The current work is a simulation-validated, software-complete proof of concept, and several limitations should be understood before it is read as a field-ready system. First, and most fundamentally, end-to-end system performance (navigation, coverage, latency, and the 62 FPS inference figure) was validated in Microsoft AirSim on an RTX 3060 desktop workstation, not on physical Jetson NX hardware; only the MobileNetV3 classification component was evaluated against real data (1,260 held-out real coconut-palm images). Physical Jetson NX benchmarking — including sustained-load thermal behaviour, actual power draw, and real RPLiDAR/GPS noise characteristics — has not been performed and is the single most important next step. Second, the five AirSim missions reuse one static 2-hectare, 400-tree scene, so they demonstrate repeatability rather than independent replication; a confusion matrix, per-class precision/recall/F1, statistical significance testing, and baseline comparison against alternative lightweight architectures (e.g., MobileNetV2, EfficientNet-Lite, ShuffleNetV2) on the same dataset have not yet been carried out, and all are necessary before the classification results can be considered fully validated. Third, the health index weights (Section V-E) reflect a design choice to emphasise visual classification confidence over LiDAR-derived structural metrics; they have not been calibrated through formal expert elicitation or validated against independent agronomic ground truth, and a weight-sensitivity analysis is needed. In addition, the tree-height range reported in the text (12-23 m) does not currently match the range plotted in Fig. 12 (~4-10 m) or the example tree in Fig. 7 (18.5 m); this inconsistency has been identified but not yet resolved, pending the authors confirming which figures reflect the correct underlying data. Fourth, the RPLiDAR provides a tilted 2D scan rather than a true 3D

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

point cloud, which limits analysis of complex canopy structures in dense plantations; a solid-state 3D LiDAR such as the Livox Mid-360, combined with volumetric models such as PointNet++ or KPConv, would improve structural assessment. Fifth, MobileNetV3 classification accuracy is expected to degrade in extreme lighting (direct sun, deep shade, fog) not represented in the training or simulation data; incorporating the NIR and Red Edge channels already present on the camera payload for true NDVI/NDRE computation may help distinguish early stress from natural senescence. Sixth, the fixed lawnmower coverage path does not adapt to dynamic obstacles or prioritise high-risk areas; a reinforcement-learning navigation policy (e.g., E-DQN, PPO) trained in AirSim could address this. Finally, coverage area, endurance, and multi-drone/multi-battery scaling have only been exercised on the single 2-hectare simulated scene; extrapolating to larger operational areas would require validation against a real battery discharge model rather than AirSim's non-physical one, and is not claimed here. Looking ahead, the priority order for future work is: (i) physical deployment on a DJI Matrice-class hexacopter with a real Jetson NX (or Jetson Orin) and RPLiDAR A3, including thermal and power benchmarking; (ii) field trials on real coconut, arecanut, and mango plantations in Karnataka across at least one growing season, with agronomist-verified ground-truth labels; (iii) a full ablation and baseline study (confusion matrices, precision/recall/F1, MobileNetV2/EfficientNet-Lite/ShuffleNetV2 comparisons, and health-index weight-sensitivity analysis); (iv) reinforcement-learning-based adaptive navigation; and (v) integration with commercial farm-management platforms for downstream irrigation and fertilisation decisions.

IX. CONCLUSION

This paper shows that the delay in traditional drone-based crop monitoring, which can take hours to days, is not a limitation of technology but a result of how the system is designed. By performing the analysis directly on the drone, this delay is completely eliminated. The proposed Edge AI system runs the full MobileNetV3 model for crop health assessment at 62 FPS on the TensorRT FP16 engine, benchmarked on the RTX 3060 simulation workstation as a proxy for the target NVIDIA Jetson NX module intended for onboard deployment. It combines visual data from the camera with structural data from the RPLiDAR sensor using the health index formula $H = 0.667 \cdot S_cam + 0.333 \cdot S_str - A$, clipped to [0, 100] (Equation 27), and prepares a detailed report about the field by the time the drone lands, with no data sent to external cloud servers during flight (telemetry and results are streamed only to the local Flask/Wi-Fi dashboard). Depthwise-separable convolution cuts the number of MobileNetV3 calculations by a factor of $1/C_out + 1/K^2$ (Equation 4), a roughly 7× MAC reduction relative to standard convolution (Section IV-A), and MobileNetV3-Large requires 18.7× fewer MACs than ResNet-50 overall. On the RTX 3060 simulation workstation, the TensorRT FP16 engine sustained 62 FPS; confirming this on physical Jetson NX hardware, together with thermal and power characterisation, is the most important remaining validation step. The system was exercised across five AirSim missions over a single simulated scene and, separately, on 1,260 held-out real coconut-palm images, meeting or exceeding all ten internally defined targets: classification accuracy of 98.2%, 92.5%, and 95.8% on real imagery; simulated LiDAR height accuracy with MAE of 14.3 cm; simulated processing speed of 62 FPS; full simulated field coverage in 18.4 minutes; and simulated Flask API, heatmap, and PDF-generation latencies of 22 ms, 187 ms, and 3.1 seconds respectively. To the best of our knowledge, this integration of real-time onboard inference, camera-LiDAR fusion, autonomous flight, live 3D visualisation, and automatic reporting within one drone platform is among the more complete such efforts reported to date, though the comparison in Table III is a capability checklist rather than a controlled benchmark. These results constitute a simulation-validated, software-complete proof of concept; physical-hardware deployment, field trials on real coconut, arecanut, and mango plantations, and formal statistical validation (Section VIII) are required before the framework can be considered ready for operational use.

REFERENCES

[1] Food and Agriculture Organization of the United Nations, "The Future of Food and Agriculture: Alternative Pathways to 2050," FAO, Rome, Italy, 2018. [Online]. Available: https://www.fao.org/3/I8429EN/i8429en.pdf

[2] D. J. Mulla, "Twenty five years of remote sensing in precision agriculture: Key advances and remaining knowledge gaps," Biosystems Engineering, vol. 114, no. 4, pp. 358–371, Apr. 2013, doi: 10.1016/j.biosystemseng.2012.08.009.

[3] A. D. Boursianis, M. S. Papadopoulou, P. Diamantoulakis, A. Liopa-Tsakalidi, P. Barouchas, G. Salahas, G. Karagiannidis, S. Wan, and S. K. Goudos, "Internet of Things (IoT) and agricultural unmanned aerial vehicles (UAVs) in smart farming: A comprehensive review," Internet of Things, vol. 18, p. 100187, May 2022, doi: 10.1016/j.iot.2020.100187.

[4] S. K. Sarma and S. K. Baruah, "Time-sensitive agricultural interventions: A review of latency impacts in UAV-based crop monitoring systems," Computers and Electronics in Agriculture, vol. 196, p. 106874, May 2022, doi: 10.1016/j.compag.2022.106874.

[5] D. Biczysko, L. Rak, and A. Wasik, "Edge computing in UAV systems: Challenges and opportunities for real-time aerial data processing," Electronics, vol. 11, no. 3, p. 393, Jan. 2022, doi: 10.3390/electronics11030393.

[6] J. A. J. Berni, P. J. Zarco-Tejada, L. Suarez, and E. Fereres, "Thermal and narrowband multispectral remote sensing for vegetation monitoring from an unmanned aerial vehicle," IEEE Transactions on Geoscience and Remote Sensing, vol. 47, no. 3, pp. 722–738, Mar. 2009, doi: 10.1109/TGRS.2008.2010457.

[7] W. H. Maes and K. Steppe, "Perspectives for remote sensing with unmanned aerial vehicles in precision agriculture," Trends in Plant Science, vol. 24, no. 2, pp. 152–164, Feb. 2019, doi: 10.1016/j.tplants.2018.11.007.

[8] C. Zhang and J. M. Kovacs, "The application of small unmanned aerial systems for precision agriculture: A review," Precision Agriculture, vol. 13, no. 6, pp. 693–712, Dec. 2012, doi: 10.1007/s11119-012-9274-5.

[9] N. H. Motlagh, M. Bagaa, and T. Taleb, "UAV-based IoT platform: A crowd surveillance use case," IEEE Communications Magazine, vol. 55, no. 2, pp. 128–134, Feb. 2017, doi: 10.1109/MCOM.2017.1600587.

[10] D. Li, R. Miao, and A. Bhattarai, "Real-time aerial weed detection using onboard deep learning for precision agriculture unmanned aerial vehicles," Computers and Electronics in Agriculture, vol. 198, p. 107098, Jul. 2022, doi: 10.1016/j.compag.2022.107098.

[11] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, Q. V. Le, and H. Adam, "Searching for MobileNetV3," in Proc. IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Republic of Korea, Oct. 2019, pp. 1314–1324, doi: 10.1109/ICCV.2019.00140.

[12] X. Chen, B. Zhou, W. Lin, and C. Zhao, "MobileNetV3-based real-time tomato disease classification for UAV-acquired crop imagery,"

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

Computers and Electronics in Agriculture, vol. 202, p. 107368, Nov. 2022, doi: 10.1016/j.compag.2022.107368.

[13] T. Hu, X. Sun, Y. Su, H. Guan, Q. Sun, M. Kelly, and Q. Guo, "Development and performance evaluation of a very low-cost UAV-LiDAR system for forestry applications," Remote Sensing, vol. 13, no. 1, p. 77, Dec. 2020, doi: 10.3390/rs13010077.

[14] Y. Chao, R. Dillmann, A. Roennau, and Z. Xiong, "E-DQN-based path planning method for drones in an AirSim simulator under a virtual agricultural environment," Biomimetics, vol. 9, no. 4, p. 238, Apr. 2024, doi: 10.3390/biomimetics9040238.

[15] Z. Al-Mania, T. R. Sheltami, G. Ahmed, A. A. Mahmoud, and A. Barnawi, "Energy-efficient online path planning for Internet of Drones using reinforcement learning," Journal of Sensor and Actuator Networks, vol. 13, no. 5, p. 50, Aug. 2024, doi: 10.3390/jsan13050050.

[16] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, Jun. 2018, pp. 4510–4520, doi: 10.1109/CVPR.2018.00474.

[17] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in Proc. IEEE/CVF CVPR, Salt Lake City, UT, USA, Jun. 2018, pp. 7132–7141, doi: 10.1109/CVPR.2018.00745.

[18] S. Lu, J. Liu, and T. Zheng, "Edge AI-based real-time pest detection and identification on agricultural UAVs using the NVIDIA Jetson platform," IEEE Access, vol. 10, pp. 43812–43823, Apr. 2022, doi: 10.1109/ACCESS.2022.3169293.

[19] H. Slimani, R. M. de Oliveira, and F. D. G. L. Carvalho, "Deep learning structure for real-time crop monitoring based on neural architecture search and unmanned aerial vehicle," Brazilian Archives of Biology and Technology, vol. 67, e24240123, Sep. 2024, doi: 10.1590/1678-4324-2024240123.

[20] X. Yang, T. Gao, Y. Han, Z. Wang, J. Liu, H. Chen, and B. Zhang, "Advances in UAV remote sensing for monitoring crop water and nutrient status: Modeling methods, influencing factors, and challenges," Plants, vol. 14, no. 16, p. 2544, Aug. 2025, doi: 10.3390/plants14162544.

[21] NVIDIA Corporation, "NVIDIA Jetson NX System-on-Module Data Sheet," Rev. 1.2, Santa Clara, CA, USA, 2023. [Online]. Available: https://developer.nvidia.com/embedded/jetson-xavier-nx

[22] Slamtec Co., Ltd., "RPLIDAR A3 — Low Cost 360 Degree Laser Range Scanner: User Manual and Interface Protocol," Document Rev. 1.0, Shanghai, China, 2022. [Online]. Available: https://www.slamtec.com/en/Lidar/A3

[23] I. Maza and A. Ollero, "Multiple UAV cooperative searching operation using polygon area decomposition and efficient coverage algorithms," in Distributed Autonomous Robotic Systems 6, R. Alami, R. Chatila, and H. Asama, Eds. Tokyo, Japan: Springer, 2007, pp. 221–230, doi: 10.1007/978-4-431-35873-2_21.

[24] R. van Essen, E. J. van Henten, and G. Kootstra, "Learning UAV-based path planning for efficient localization of objects using prior knowledge," arXiv preprint arXiv:2412.11717, Dec. 2024. [Online]. Available: https://arxiv.org/abs/2412.11717

[25] L. Wallace, A. Lucieer, C. Watson, and D. Turner, "Development of a UAV-LiDAR system with application to forest inventory," Remote Sensing, vol. 4, no. 6, pp. 1519–1543, Jun. 2012, doi: 10.3390/rs4061519.

[26] A. T. T. Phan and K. Takahashi, "Estimation of rice plant height from a low-cost UAV-based LiDAR point cloud," International Journal of Geoinformatics, vol. 17, no. 2, 2021.

[27] T.-M. Hoang, V.-H. Bui, V.-S. Nguyen, D.-T. Doan, H.-A. Dang, and A.-T. Pham, "A comprehensive evaluation of lightweight deep learning models for tomato disease classification on edge computing environments," Scientific Reports, vol. 16, art. 12320, Mar. 2026, doi: 10.1038/s41598-026-42439-6.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)