

Offline Forest Pest Recognition Using Edge Computing and Lightweight Neural Networks

Ke Jin, Yuxuan Zhong, Bo Peng

School of Mechanical and Vehicle Engineering, Changchun University, Changchun, China

Corresponding Author: Ke Jin. Email: 012140817@mails.ccu.edu.cn

ABSTRACT: Forest patrol requires close-range recognition of biological threats under variable illumination, limited connectivity, and restricted computing resources. This paper presents an offline recognition system that combines a local visual recognition application with a wearable edge-computing architecture for forest pest monitoring and patrol support. The study reconstructs the system from technical documentation, application source code, datasets, and archived training records. It describes image acquisition, confidence-based recognition, contextual event recording, and the proposed connection to a local decision-support knowledge base. A reproducibility audit identifies two distinct data resources: a four-class plant-disease classification dataset containing 5,932 image files and a sixteen-category forest-pest annotation dataset containing 5,249 image–annotation pairs. The archived classifier log records a top-1 validation score of 1.000 at the final epoch, but 315 unique image hashes occur in both classification splits. Therefore, this score cannot establish independent field accuracy or forest-pest detection performance. The system report separately gives a mean inference latency of 218 ms and an end-to-end latency of 352 ms; these are treated as reported prototype measurements rather than independently reproduced benchmarks. The principal contribution is an explicit engineering account of the offline patrol workflow, together with a traceable distinction between implemented recognition functions, reported hardware capabilities, and requirements for further validation. The analysis supports continued development of local visual assistance while identifying dataset independence, model–label consistency, uncertainty handling, and sustained device testing as prerequisites for reliable deployment.

Keywords: forest pest recognition; edge computing; offline patrol; image classification; wearable monitoring; reproducibility; decision support.

Date of Submission: 09-10-2026

Date of acceptance: 20-10-2026

NOMENCLATURE

Symbol	Description	Unit
K	Number of recognition categories	—
p	Model class score	—
tau	Acceptance threshold	—
t _{inf}	Model inference interval	ms
t _{e2e}	Capture-to-output interval	ms
P	Electrical load	W
E _{use}	Usable battery energy	Wh

I. INTRODUCTION

Forest health monitoring depends on detecting biological threats early enough for specialists to investigate and managers to respond. Insects, pathogens, drought, and other disturbance agents interact with environmental conditions, so a visible change in a tree cannot be interpreted reliably from appearance alone. Seidl et al. describe how climate change affects forest disturbance regimes and the relationships among disturbance agents [1]. For ground patrol, this creates a practical requirement: personnel need assistance in collecting and interpreting evidence without losing the surrounding ecological context. A useful recognition system should support inspection, documentation, and referral rather than convert every visual prediction into a definitive diagnosis.

Computer vision offers a route to more consistent image-based screening. Work on plant disease recognition demonstrates that neural networks can learn diagnostic visual patterns from labeled images [2], while research using field photographs of cassava illustrates the relevance of transfer learning to agricultural diagnosis [3]. Nevertheless, a forest patrol image differs from a carefully framed laboratory sample. A small

insect may occupy only a few pixels; branches and leaves may obscure it; motion, glare, and background texture can hide the features required for recognition. A model trained on isolated subjects may therefore perform differently when used by a moving operator beneath a forest canopy.

Connectivity introduces a second constraint. Uploading every frame to a remote server makes recognition dependent on network availability and introduces an additional communication step. Local processing is attractive where patrol routes extend into weak-signal areas. Edge computing places computation close to the source of observations and shifts attention toward resource management and deployment constraints [4]. In a wearable patrol device, those constraints include battery energy, heat dissipation, storage reliability, and the operator's ability to view a result while walking. Offline processing can address communication dependence, but it does not by itself guarantee accurate predictions, short response times, or secure storage.

The proposed system addresses this problem through a wearable edge-computing architecture. The technical documentation describes a smart helmet intended to connect camera observations with recognition, positioning, environmental information, local advice, and event storage [5]. The submitted software archive provides a desktop application for images, image folders, video, and camera input [6]. These materials capture both the practical recognition interface and a broader wearable-system design. They also contain different model descriptions and evaluation records, making it necessary to establish which conclusions each artifact can support before translating the system into a research manuscript.

This paper has three objectives. First, it specifies a coherent offline patrol workflow grounded in the supplied engineering materials. Second, it examines the actual application and training artifacts to clarify the task being implemented. Third, it evaluates the strength and limitations of the available evidence, including dataset composition and cross-split overlap. The study is an engineering reconstruction and artifact-based evaluation; it does not introduce a new detector or claim that a fresh field trial was conducted. Its contribution lies in joining the application architecture to a reproducible evidence account and defining the validation steps needed to move from a demonstrator to a dependable patrol tool.

II. RELATED WORK

2.1 Image recognition for pests and plant disease

Plant disease classification has shown that supervised visual learning can provide useful screening signals when labels and image acquisition are well defined [2, 3]. However, classification assigns a category to an image, whereas detection identifies the locations and categories of individual objects. This distinction matters in a patrol scene containing several insects or several possible symptoms. A classifier may respond to the most visually dominant subject without identifying the target that concerns the operator. The IP102 benchmark explicitly distinguishes insect-pest classification from bounding-box detection and documents variation within and between categories as well as class imbalance [7]. These challenges are directly relevant to a forest dataset that represents eggs, larvae, pupae, and adults as separate labels.

Single-stage detectors offer one possible approach to localization. YOLO predicts spatial detections through a unified network [8], and SSD uses multiscale feature maps and default boxes to recognize objects at different sizes [9]. Focal loss was proposed to address the imbalance between easy and hard examples in dense detection [10]. These works motivate design choices for future forest-scene detection, but their published benchmark results cannot be transferred numerically to the present system. Differences in species, object size, annotation quality, input resolution, and hardware prevent a meaningful direct performance ranking without experiments on a common evaluation set.

2.2 Efficient neural networks and compression

The choice of visual backbone influences deployment cost. Residual learning provides a widely used basis for image representation [11]. MobileNets use depthwise separable convolutions to reduce computation [12], MobileNetV2 introduces inverted residuals and linear bottlenecks [13], and MobileNetV3 combines architecture design with hardware-aware optimization [14]. EfficientNet examines joint scaling of depth, width, and input resolution [15]. These approaches show that efficiency must be considered throughout the model architecture. A nominally small checkpoint is not sufficient evidence of fast execution: runtime kernels, memory movement, and the supported numerical precision also determine practical latency.

Compression techniques provide additional options. Deep Compression combines pruning, quantization, and coding to reduce network storage [16]. Filter pruning removes complete convolutional filters [17], while network slimming learns channel-level sparsity that can be converted into smaller dense networks [18]. Knowledge distillation transfers information from a teacher to a student model [19], and integer-only quantization connects low-precision training and inference [20]. The system report mentions several of these strategies [5]. In the present study, they are described as the reported optimization route and as candidates for controlled future ablation, because the archive does not provide the full sequence of teacher checkpoints,

pruning configurations, calibration samples, and exported integer models needed to reproduce the claimed compression gains.

2.3 Confidence, rejection, and operational interpretation

A recognition score should not be interpreted automatically as the probability that a biological diagnosis is correct. Calibration research shows that modern neural networks can produce confidence values that do not match empirical correctness frequencies [21]. Out-of-distribution detection research further indicates that unfamiliar inputs require deliberate evaluation rather than reliance on the largest softmax score [22]. A patrol system will encounter healthy trees, unrelated insects, blurred frames, equipment, and objects absent from training. The interface must therefore support uncertainty and referral. These concerns connect visual modeling to the design of local advice: downstream actions should reflect the quality and scope of the visual evidence.

III. MATERIALS AND METHODS

3.1 Study materials and evidence categories

The study uses the Chinese-language technical report and the accompanying application archive [5, 6]. The report is the source for the wearable architecture, peripheral descriptions, and prototype timing statements. The source files provide evidence of image classification, confidence display, batch processing, camera handling, and video handling. Dataset directories and XML annotations provide evidence of sample counts and label vocabulary. Archived CSV records and plot files provide evidence of the metrics logged during a previous classification run. No new model training, checkpoint execution, field deployment, or destructive modification of the original datasets was performed for this manuscript.

Evidence is separated into three categories. Artifact-observed findings are derived directly from file contents, directory inventories, annotation parsing, and byte-level image hashing. Reported findings are statements or measurements reproduced from the technical report with their source identified. Proposed requirements describe implementation or validation work that would strengthen the system. This separation prevents a design diagram from being treated as proof of integration and prevents an archived training score from being described as an independently reproduced field result. It also permits useful system analysis even where the archive is incomplete.

3.2 Dataset inventory and class scope

Two visually relevant datasets coexist in the archive. The directory named Data is organized for classification and contains four labels: Bacterialblight, Blast, Brownspot, and Tungro. Its training split contains 4,749 image files and its validation split contains 1,183 image files. A separate directory named Linye, represented by the Chinese folder name for forestry, contains 4,097 training images, 654 validation images, and 498 test images. Each image in this second resource has an XML annotation file. Parsing the object-name elements identifies sixteen distinct annotation categories. Table 1 summarizes the inventory; the counts refer to files, not to unique biological specimens or independent acquisition sessions.

Table 1. Inventory of the supplied datasets; image counts do not establish independent observations.

Resource and split	Image files	XML files	Task / categories
Data / train	4,749	0	Classification / 4
Data / val	1,183	0	Classification / 4
Data / total	5,932	0	Classification / 4
Forestry / train	4,097	4,097	Object annotations / 16
Forestry / valid	654	654	Object annotations / 16
Forestry / test	498	498	Object annotations / 16
Forestry / total	5,249	5,249	Object annotations / 16

The forest-pest labels include *Drosicha contrahens* females, *Erthesina fullo* nymphs, *Phenacoccus*, *Sericinus montela* larvae, *Cnidocampa flavescens* pupae, and an annotation category named citricola scale. Additional labels cover *Latoia consocia* larvae; *Plagiodera versicolora* larvae, eggs, and another life-stage-unspecified category; *Spilartia subcarnea* larvae; *Cerambycidae* larvae; *Micromelalopha troglodyta* larvae; *Hyphantria cunea* larvae and pupae; and aphids. The names are reported as the archive's vocabulary rather than a newly verified taxonomic identification. Species, life stage, and symptom categories should remain separate fields in future dataset documentation so that label meaning does not depend on informal naming conventions.

The technical report gives several larger dataset totals in different sections, including more than 240,000 original images, approximately 500,000 images after augmentation, and more than 50,000 training

images [5]. Those descriptions cannot be reconciled with the smaller distributed archive from file counts alone. A partial archive is one possible explanation, but the missing images are not available to verify it. Accordingly, the analysis uses the observed counts in Table 1 and does not claim access to the larger reported corpus. The four-class disease resource is also not interchangeable with the sixteen-category forest-pest resource merely because both concern plant health.

3.3 Cross-split integrity audit

Every image file in the two dataset resources was read and assigned a SHA-256 digest. For each split, a set of distinct digests was constructed. The intersection between two split sets identifies byte-identical content stored across those splits. This method is deterministic and detects exact file duplication without requiring a trained model. It does not detect all visually similar derivatives: recompression, cropping, color modification, or other augmentation may change a digest even when two files originate from the same photograph. Therefore, a zero intersection should be interpreted as the absence of byte-identical overlap, rather than proof of independent sampling.

The classification training split contains 3,987 distinct image hashes, and the validation split contains 1,122 distinct hashes. Their intersection contains 315 hashes. Relative to distinct validation content, the overlap is 315/1,122, or approximately 28.1%. The forest-pest splits contain 4,097, 654, and 498 distinct hashes, respectively, and no exact hash overlap occurs between any pair. A secondary check compares filename prefixes before the .rf. marker, where present. It finds 177 shared prefixes between forest training and validation, 170 between training and test, and 39 between validation and test. These filename matches are screening indicators of possibly related derivatives; provenance records and visual inspection are needed before declaring them confirmed leakage.

Table 2. Cross-split integrity audit. Filename-prefix matches are screening indicators, not confirmed source identity.

Resource	Split comparison	Shared exact hashes	Shared filename prefixes
Data	Train / val	315	0
Forestry	Train / valid	0	177
Forestry	Train / test	0	170
Forestry	Valid / test	0	39

An independent split should be formed at the source-observation level. If multiple views belong to one insect, tree, patrol visit, or image-augmentation family, the group should be assigned wholly to a single split before augmentation. Otherwise, performance can reflect familiarity with the original scene rather than recognition of unseen biological cases. Exact hashing should be combined with acquisition metadata, augmentation lineage, and a review of near-duplicate images. The current audit establishes a specific problem in the classification resource and a need for provenance review in the forest resource; it does not estimate the size of any resulting accuracy inflation.

3.4 Archived training configuration and metric interpretation

The training entry point initializes the Ultralytics YOLOv8 nano classification model and requests 250 epochs with batch size four. The archived argument file records the classification task, a 224-pixel input setting, pretrained initialization, seed zero, deterministic execution, and automatic optimizer selection. Official Ultralytics documentation distinguishes YOLOv8 task variants and their outputs [23, 24]. The recorded optimizer option is auto, so this manuscript does not assign a specific optimizer implementation without the corresponding runtime log. The stored configuration references the Data directory and a classification save path, which is essential context when interpreting the metrics.

The archived CSV has 250 epoch rows. It records a first-epoch top-1 validation score of 0.85714 and a final-epoch score of 1.000. The first logged score of 1.000 occurs at epoch six. Training loss falls from 0.0652 to 0.00001, while validation loss changes from 0.12153 to 0.09289. These quantities are observations from an existing log, not the results of a rerun. Because the file sits under a general runs/train directory while its argument file refers to a different classification save directory, the original run should ultimately be linked through a manifest containing configuration, checkpoint, software version, dataset hashes, and execution logs.

Top-1 accuracy measures the fraction of images whose selected class agrees with the image label. It does not measure localization quality. Detection evaluation additionally requires a rule for matching predicted boxes to annotated objects, including an intersection-over-union threshold. COCO illustrates a benchmark design centered on object annotations and detection evaluation [25]. Consequently, a classification accuracy of 1.000 cannot be renamed mAP@0.5. Top-5 accuracy is also uninformative for a four-class problem when all

available classes are included among the candidate predictions. These distinctions are necessary to avoid making the archived run appear to validate the system's broader sixteen-category patrol task.

IV. SYSTEM DESIGN AND IMPLEMENTATION

4.1 Offline patrol architecture

The intended architecture follows an observation-to-record sequence. A camera captures a view selected during patrol; local preprocessing prepares the image for inference; the model produces a category or, in a future detection branch, localized candidates. A confidence and validity layer determines whether the result can support an advisory message. Position, time, and environmental measurements are then attached to an event record, and a local knowledge component selects an appropriate inspection or reporting action. The event is displayed and stored for later transfer at the patrol station. This design keeps the recognition path available without depending on an external inference service.

The system report identifies a Raspberry Pi Compute Module 4 with 4 GB memory as the intended computing platform, together with a camera, display, positioning module, and environmental peripherals [5]. The manufacturer describes Compute Module 4 as an embedded computing module with interfaces suitable for custom carrier designs [26]. That hardware choice provides an integration basis, but suitability for a particular model must be established by testing the complete runtime. The supplied photographs document a physical prototype presentation; they do not independently establish the processor configuration, waterproofing performance, positioning accuracy, or battery endurance of the assembled device.

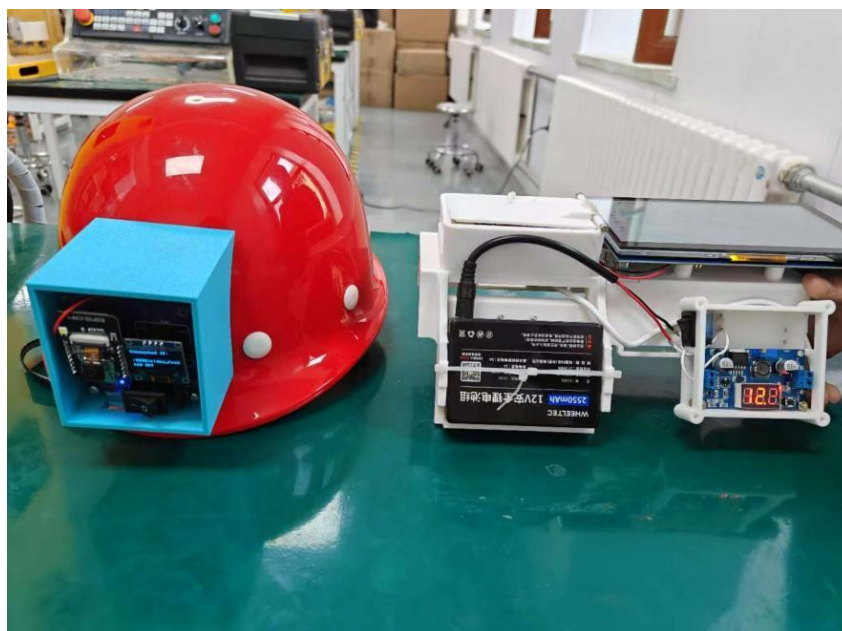


Fig. 1. Helmet and terminal prototype shown in the supplied system report. The photograph documents the system presentation, not an independent deployment test.

4.2 Implemented visual recognition interface

The application uses PyQt5 for its graphical interface, OpenCV for image and video handling, and the Ultralytics interface for model loading [6]. During initialization, the code selects a GPU when CUDA is available and otherwise selects the CPU. It loads models/best.pt with task set to classify. The interface connects controls for a single image, a folder of images, video input, and camera input. Image predictions are extracted from the result's probability vector and top-1 index. The code displays a class name, confidence text, processing time, and an entry in the result table. This demonstrates the structure of a local image-classification interface, rather than a verified object-localization pipeline.

For accepted predictions, the displayed Chinese name is selected from an ordered sixteen-entry list in the configuration file. A separate English-name collection is defined as a set. Since a set does not constitute a stable ordered class-index mapping, it should not serve as the source of model label order. More fundamentally, the displayed list must match the labels stored with the checkpoint that is actually loaded. The four-class training resource and sixteen-name user interface coexist in the archive, but the inspected files do not prove that the deployed checkpoint came from that four-class run. A checkpoint-bound mapping and startup consistency check would resolve this uncertainty without assuming that every stored model artifact belongs to the same experiment.

The interface uses a confidence threshold of 0.3. Above that value, the selected class is shown; below it, the interface displays an unrecognized label. In the inspected image-processing branch, the confidence text for this rejected state is set to 100%. That presentation should be interpreted as a user-interface inconsistency, not as evidence of certain recognition of an unknown class. A clearer representation would display the actual prediction score and an explicit abstention state. The threshold should be selected using validation data that include unfamiliar insects, healthy backgrounds, and low-quality images, rather than fixed solely because it produces more accepted predictions.

4.3 Scheduling and operator interaction

The source defines a detection frequency of one prediction every five frames for video and camera operation. This can reduce the number of inference calls, but its effect depends on capture rate, buffering, and event-loop behavior. A patrol terminal needs to avoid a growing queue of old frames; a fast-looking preview is not useful if its recognition label refers to an earlier scene. A latest-frame policy is a suitable candidate for implementation, with timestamps retained for both acquisition and inference completion. Capture, inference, display refresh, and event persistence should be measured separately so that bottlenecks can be located without confusing display frame rate with recognition throughput.

User interaction should support evidence gathering with minimal interruption. A worker needs to confirm which subject was analyzed, capture a sharper image when necessary, and retain an uncertain observation for expert review. Audible notifications can be useful, but repeated predictions of the same object should not produce continuous alarms or duplicate incident records. Temporal confirmation, a short notification cooldown, and explicit user confirmation are possible mechanisms to evaluate. These are design recommendations derived from the patrol workflow; the current desktop source inspection does not establish that they have been implemented in the wearable prototype.

4.4 Contextual records and local advice

The report proposes combining visual results with location, environmental readings, and time [5]. A practical event record should contain an event identifier, original image reference, model version, predicted label, confidence score, inference task, timestamp, and operator confirmation status. Location needs both coordinates and a quality indicator, since canopy cover can reduce satellite reception. Environmental values should retain their units and collection time. Keeping these fields separate makes it possible to distinguish an actual sensor measurement from a value inferred later. It also allows managers to investigate an event even if one peripheral was unavailable during capture.

Local advice can be organized as versioned condition–action rules. A rule might refer a suspected pest to specialist review and request additional photographs when the result is uncertain. It should state the evidence required, the region and season in which it applies, and the responsible knowledge source. Advice must remain an aid to inspection rather than a substitute for identification methods that require specimens or laboratory analysis. The report's examples of specific pest-control actions are therefore not adopted as universally valid biological rules. No claim is made here that the inspected desktop source independently demonstrates a complete positioning, encrypted event-storage, or rule-inference subsystem.

Offline updates require the recognition model, class mapping, preprocessing specification, and rule version to remain compatible. An update bundle should identify these components together, include integrity checks, and support recovery to a previously working version. Merely copying a new checkpoint to the device can change output indices and break the displayed labels or downstream rules. The system materials describe physical-media updates as an intended maintenance route [5]. The engineering requirement is to make that route testable: operators should be able to verify the installed version, run a known sample, and recover when an update is interrupted or incompatible.

V. RESULTS AND DISCUSSION

5.1 Artifact-based findings

The archive supports a narrower but concrete set of findings. A local visual classification application has been specified in executable source, a sixteen-entry user-facing pest vocabulary is present, and two dataset resources are supplied. Their inventories and exact-duplication properties can be reconstructed without training a model. The stored learning curves and CSV describe a classification experiment that converged rapidly on its validation split. Figure 2 reproduces the archived plot for that run. It is labeled as classification evidence and should not be interpreted as a new forest detection experiment or a comparison among compressed model variants.

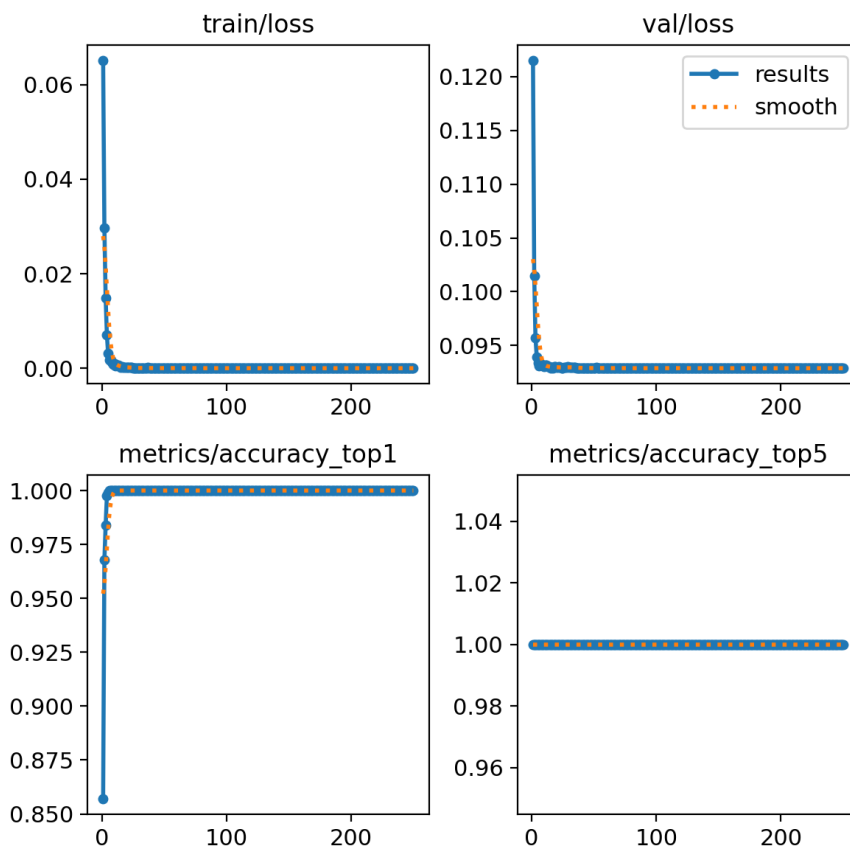


Fig. 2. Archived learning curves for the four-class classification run. The validation score is affected by the split-overlap limitation identified in this study.

The final validation score of 1.000 is numerically consistent with the archived plot, but its meaning is limited by the exact duplication identified in Section III. The validation split contains both unique material and content identical to training images. Without reevaluation on an independent split, the score cannot quantify generalization to new patrol scenes. The audit does not prove that all predictions resulted from memorization, nor does it calculate a corrected score. Its supported conclusion is that the available validation result is insufficient as evidence of independent accuracy. This finding changes how the system should be evaluated, rather than negating the value of the implemented interface.

The forest annotation resource is more closely aligned with the interface vocabulary and contains a distinct test split. Its absence of byte-identical overlap is encouraging at the file-content level. However, common filename prefixes suggest that some transformed samples may share an original source across splits. Moreover, the available classification log evaluates a different four-label resource. The forest images therefore provide material for a future task-aligned experiment, but they do not retroactively establish the performance of the archived classifier or prove that the current checkpoint recognizes every forest label correctly.

5.2 Reported latency and its interpretation

The technical report gives a mean model-inference time of 218 ms with a standard deviation of 15 ms, an end-to-end processing time of 352 ms with a standard deviation of 22 ms, and a decision-generation time below 10 ms [5]. It states that timing was performed after one hour of continuous operation. Table 3 preserves these values and distinguishes their source. Raw timestamp sequences, sample counts, software versions, and a deployment manifest are not supplied, so the reported standard deviations cannot be converted into confidence intervals or independently verified. Earlier passages mention shorter inference targets; the explicit timing table is used here to avoid mixing target values with reported measurements.

Table 3. Prototype timing reported in the technical report; no new timing experiment was performed.

Stage	Reported mean	Reported SD	Evidence
Model inference	218 ms	15 ms	Report timing table
End-to-end processing	352 ms	22 ms	Report timing table
Advice generation	<10 ms	Not supplied	Report timing table

Using the reported means, inference occupies approximately 218/352, or 61.9%, of the end-to-end interval. The remaining difference is 134 ms and includes stages outside pure model inference, although the archive does not assign that difference to individual components. The reciprocal of 0.352 s is approximately 2.84 processed images per second for a strictly sequential pipeline. This is an arithmetic interpretation, not a measured sustained camera throughput. Frame skipping, asynchronous capture, display refresh, storage writes, and thermal behavior can change the realized rate. The result is relevant to a deliberate close-range inspection workflow, but it should not be described as high-frame-rate detection of every object encountered while walking.

The latency figures suggest two optimization directions. Model compression may reduce the inference component, while preprocessing, synchronization, and persistence may account for part of the remaining delay. A smaller model can still leave substantial non-model overhead, and an integer export can introduce conversion or kernel overhead when the runtime does not support it efficiently. The compression literature motivates these options [16, 17, 18, 19, 20], but a useful system-specific comparison requires a baseline and each modified model to be tested under the same capture workload, device temperature, input size, thread count, and numerical precision.

5.3 Accuracy and deployment claims in the report

The report describes an independent 50-image test sample comprising 20 newly captured field images, 20 selected public images, and additional difficult negatives [5]. It states mAP above 90% and F1 above 89% in one passage, while later passages give stronger summary values. The supplied materials do not include a complete per-image prediction file, classwise ground truth, box-matching definition, or common metric table that reconciles these statements. This manuscript therefore reports the existence of those preliminary claims without treating their strongest value as a verified result. A detection score requires documented annotations and matching rules; a classification score requires image labels and a consistent decision rule.

A small test set can be useful for detecting major integration failures, but it does not characterize the full diversity of forest conditions. Images from one visit may share background, camera settings, and specimen identity. Rare pest stages may have few examples, and healthy negatives may be underrepresented. Future evaluation should document the number of observations per class and scene group, report classwise errors, and include uncertainty intervals. The system's practical value is likely to depend particularly on missed targets and incorrect high-confidence referrals, rather than only on one pooled percentage.

The source report also estimates runtime by dividing a 10,000 mAh battery capacity by a 1.2 A operating current, yielding about 8.3 hours [5]. This calculation requires both quantities to refer to the same voltage basis and does not establish measured endurance. The stated 5 V operating point corresponds to approximately 6 W of load. Battery assessment should instead use usable energy in watt-hours, account for conversion losses, and measure operation until the device's actual cutoff condition. Because battery-cell voltage and usable energy are not documented in the archive, an all-day endurance claim is not independently supported by the ratio alone.

5.4 Reliability boundaries and practical value

The most defensible near-term role is supervised screening and evidence collection. An operator can capture an image, inspect the proposed category, save the observation, and seek confirmation when needed. This workflow can be useful even before full automatic event localization and decision support are integrated. Its value comes from consistent recording and rapid access to a provisional visual interpretation. A field experiment should quantify that value through task completion time, record completeness, referral quality, and operator workload, rather than assume that local inference automatically improves forest management outcomes or reduces control costs by a fixed percentage.

Confidence errors, label mismatches, and stale frames can undermine this workflow even when average classification accuracy is high. The 100% text used for a rejected prediction is an example of how interface semantics can contradict model semantics. A wrong index-to-name mapping can produce a fluent but incorrect label. Missing location quality can make a point appear precise when reception was poor. These failure modes are not all solved by retraining. They require explicit software contracts and tests connecting model outputs, displayed names, contextual metadata, and stored records.

VI. VALIDATION AND DEVELOPMENT ROADMAP

The first priority is a reproducible data and model manifest. Each sample should carry a source identifier, acquisition context, annotation version, and augmentation lineage. Source groups must be separated before augmentation, and the byte-duplicate classification overlap must be removed from any independent evaluation. The classification and forest-detection tasks should have separate manifests, checkpoints, and metric reports. This would prevent accidental mixing of a four-class plant-disease score with a sixteen-label forest

interface. Evaluation should then use a held-out forest-patrol collection gathered at new sites or visits, with enough examples of each relevant life stage to identify class-specific failure patterns.

The second priority is a controlled model comparison. A task-aligned baseline should be evaluated before applying structured pruning, distillation, or quantization. Each modification should be accompanied by the exported artifact, parameter or storage measurements, and performance on the same independent test set. Mobile backbones and efficient scaling approaches provide reasonable alternatives to compare [12, 13, 14, 15]. However, the purpose of comparison is to identify the best tradeoff for the patrol workload, rather than reproduce headline results from unrelated benchmarks. Threshold selection and uncertainty evaluation should use a separate calibration set, with both accepted and rejected cases reported [21, 22].

The third priority is sustained hardware testing. The complete terminal should process a repeatable camera workload while logging capture times, inference completion, display updates, storage events, processor temperature, and power. Median and tail latency should accompany the mean, and dropped or stale frames should be counted. Battery endurance should be measured under a declared screen brightness and peripheral configuration. Positioning should be assessed with reference points under both open sky and canopy conditions. Where signal quality is inadequate, the interface should preserve the uncertainty rather than report a universal sub-meter location accuracy. Environmental durability also needs explicit test conditions before protective performance is claimed.

Finally, the advisory subsystem should be evaluated as a separate component. Forestry specialists should review the rule vocabulary, applicable regions, and escalation logic. Tests should cover missing sensors, unknown labels, low-confidence images, conflicting rules, and incompatible updates. A practical field study can compare the assisted workflow with the existing patrol procedure using the same inspection tasks. Measures should include the quality of incident records and the correctness of referrals, together with operator comfort and interruption burden. These steps would connect algorithmic performance to actual patrol effectiveness while preserving human responsibility for diagnosis and treatment decisions.

VII. CONCLUSION

The offline recognition system addresses a practical need for local visual assistance during forest patrol in areas with limited connectivity. The supplied materials support an engineering architecture that connects image acquisition, recognition, contextual documentation, and proposed local advice, together with a desktop implementation of image classification. The archive contains a four-class disease resource with 5,932 image files and a sixteen-category forest-pest resource with 5,249 image–annotation pairs. A direct integrity audit finds 315 shared unique image hashes between the classification training and validation splits, limiting interpretation of the recorded final top-1 score of 1.000. The report's 218 ms inference and 352 ms end-to-end means remain useful preliminary timing statements, but they require raw records and a matched deployment manifest for independent reproduction.

The system should therefore be understood as a promising recognition and patrol-support prototype with clearly defined validation needs. Its current contribution is the integration concept and the observable recognition workflow, not an established superiority claim over alternative models or field practices. Reliable progression requires source-grouped dataset separation, checkpoint-bound class names, honest uncertainty display, task-specific evaluation, and sustained testing of the assembled terminal. With those conditions satisfied, an offline system can be evaluated fairly as an aid to collecting timely, interpretable evidence for forest-health management.

DATA AND SOFTWARE AVAILABILITY

The study materials were supplied as a local system archive. They are identified in references [5] and [6] but have no verified public repository or dataset license in the inspected folder. Counts, duplicate intersections, and log values reported here were derived from that supplied version. The archive should be accompanied by an exportable manifest and permission information before public redistribution. No additional human-subject dataset or new field experiment was created for this study.

REFERENCES

- [1] Seidl, R., Thom, D., Kautz, M., et al. (2017). Forest disturbances under climate change. *Nature Climate Change*, 7, 395–402. <https://doi.org/10.1038/nclimate3303>
- [2] Mohanty, S. P., Hughes, D. P., and Salathé, M. (2016). Using deep learning for image-based plant disease detection. *Frontiers in Plant Science*, 7, 1419. <https://doi.org/10.3389/fpls.2016.01419>
- [3] Ramcharan, A., Baranowski, K., McCloskey, P., Ahmed, B., Legg, J., and Hughes, D. P. (2017). Deep learning for image-based cassava disease detection. *Frontiers in Plant Science*, 8, 1852. <https://doi.org/10.3389/fpls.2017.01852>
- [4] Shi, W., Cao, J., Zhang, Q., Li, Y., and Xu, L. (2016). Edge computing: Vision and challenges. *IEEE Internet of Things Journal*, 3(5), 637–646. <https://doi.org/10.1109/JIOT.2016.2579198>
- [5] Technical documentation of an offline forest pest recognition and patrol-support system. (2026). Unpublished Chinese-language development report supplied with the source materials; dated 13 April 2026.
- [6] Forest pest recognition application source, classification training records, and dataset archive. Unpublished local software and data; inspected 11 October 2026. Files include `MainProgram.py`, `Config.py`, `train.py`, `runs/train/results.csv`, and the Data and forestry dataset directories.
- [7] Wu, X., Zhan, C., Lai, Y.-K., Cheng, M.-M., and Yang, J. (2019). IP102: A large-scale benchmark dataset for insect pest recognition. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 8787–8796. https://openaccess.thecvf.com/content_CVPR_2019/html/Wu_IP102_A_Large-Scale_Benchmark_Dataset_for_Insect_Pest_Recognition_CVPR_2019_paper.html
- [8] Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You only look once: Unified, real-time object detection. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 779–788. <https://arxiv.org/abs/1506.02640>
- [9] Liu, W., Anguelov, D., Erhan, D., et al. (2016). SSD: Single shot MultiBox detector. *European Conference on Computer Vision*, 21–37. https://doi.org/10.1007/978-3-319-46448-0_2
- [10] Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollár, P. (2017). Focal loss for dense object detection. *Proceedings of the IEEE International Conference on Computer Vision*, 2980–2988. <https://arxiv.org/abs/1708.02002>
- [11] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 770–778. <https://arxiv.org/abs/1512.03385>
- [12] Howard, A. G., Zhu, M., Chen, B., et al. (2017). MobileNets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*. <https://arxiv.org/abs/1704.04861>
- [13] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., and Chen, L.-C. (2018). MobileNetV2: Inverted residuals and linear bottlenecks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 4510–4520. <https://arxiv.org/abs/1801.04381>
- [14] Howard, A., Sandler, M., Chu, G., et al. (2019). Searching for MobileNetV3. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 1314–1324. <https://arxiv.org/abs/1905.02244>
- [15] Tan, M., and Le, Q. V. (2019). EfficientNet: Rethinking model scaling for convolutional neural networks. *Proceedings of the 36th International Conference on Machine Learning*, PMLR 97, 6105–6114. <https://proceedings.mlr.press/v97/tan19a.html>
- [16] Han, S., Mao, H., and Dally, W. J. (2016). Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding. *International Conference on Learning Representations*. <https://arxiv.org/abs/1510.00149>
- [17] Li, H., Kadav, A., Durdanovic, I., Samet, H., and Graf, H. P. (2017). Pruning filters for efficient ConvNets. *International Conference on Learning Representations*. <https://arxiv.org/abs/1608.08710>
- [18] Liu, Z., Li, J., Shen, Z., Huang, G., Yan, S., and Zhang, C. (2017). Learning efficient convolutional networks through network slimming. *Proceedings of the IEEE International Conference on Computer Vision*, 2736–2744. <https://arxiv.org/abs/1708.06519>
- [19] Hinton, G., Vinyals, O., and Dean, J. (2015). Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*. <https://arxiv.org/abs/1503.02531>
- [20] Jacob, B., Kligys, S., Chen, B., et al. (2018). Quantization and training of neural networks for efficient integer-arithmetic-only inference. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2704–2713. <https://arxiv.org/abs/1712.05877>
- [21] Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. (2017). On calibration of modern neural networks. *Proceedings of the 34th International Conference on Machine Learning*, PMLR 70, 1321–1330. <https://arxiv.org/abs/1706.04599>
- [22] Liang, S., Li, Y., and Srikant, R. (2018). Enhancing the reliability of out-of-distribution image detection in neural networks. *International Conference on Learning Representations*. <https://arxiv.org/abs/1706.02690>
- [23] Ultralytics. Explore Ultralytics YOLOv8. Official model documentation. <https://docs.ultralytics.com/models/yolov8/> (accessed 11 October 2026).
- [24] Ultralytics. Image classification with Ultralytics YOLO. Official task documentation. <https://docs.ultralytics.com/tasks/classify/> (accessed 11 October 2026).
- [25] Lin, T.-Y., Maire, M., Belongie, S., et al. (2014). Microsoft COCO: Common objects in context. *European Conference on Computer Vision*, 740–755. <https://arxiv.org/abs/1405.0312>
- [26] Raspberry Pi Ltd. Raspberry Pi Compute Module 4. Official product documentation. <https://www.raspberrypi.com/products/compute-module-4/> (accessed 11 October 2026).