WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal Homepage: www.ijrpr.com ISSN 2582-7421

On-Board Edge AI Processing for Autonomous UAVs: A Dual-Hardware Approach for Inspecting Power Line Infrastructure

L. Sai Chaitanya[1], R. Desowja[1], P. Devi Priya[1], V. Sahiti[1], M. Purnachandra[1]*, D. Praveen Kumar[1], Dr. K. Karthick[1], Dr. R. Ramakrishna[1]

[1]Department of Electrical and Electronics Engineering, GMR Institute of Technology, Rajam, Andhra Pradesh, India
***Email: 24345A0211@gmrit.edu.in**

ABSTRACT

Power transmission infrastructure requires periodic inspection to identify damaged or degraded components. Manual inspection is time-consuming and can expose personnel to difficult and hazardous environments. This paper presents a UAV-oriented power-line asset inspection framework that combines deep-learning-based object detection, hierarchical condition classification, and onboard edge computing. The detection stage is revised to use the lightweight YOLO11n model to localize power-line assets in aerial imagery, while an EfficientNetB0-based hierarchical classifier assesses the condition of selected component types. The detection dataset is based on InsPLAD and is partitioned by transmission-tower identity to reduce visual data leakage between training, validation, and test sets. The supplied YOLO11n training plots compare 30-, 50-, and 100-epoch runs and include training/validation loss curves, mAP curves, and confusion matrices. The curves show progressive convergence, with the 100-epoch run continuing to improve after the shorter runs. The supplied confusion matrix indicates strong diagonal responses for several asset categories, while lower normalized values for some shackle-related categories indicate that class-wise performance is not uniform. Numerical YOLO11n precision, recall, mAP, inference latency, and edge throughput are not stated in the supplied graph files and are therefore not claimed in this manuscript. The system architecture retains a Raspberry Pi 5 companion computer, camera acquisition, database logging, and a web monitoring interface. Pixhawk-based autonomous flight control remains future work.

Keywords: UAV inspection; power-line assets; YOLO11n; EfficientNetB0; edge computing; InsPLAD; fault classification; Raspberry Pi 5.

1. Introduction

Overhead transmission networks contain insulators, conductors, suspension hardware, and other components that operate under electrical, mechanical, and environmental stress. Degradation of these assets can contribute to outages and maintenance costs. Conventional inspection methods depend on ground patrols, manual image review, or crewed aerial surveys. These approaches can be slow, expensive, and difficult to apply consistently across long or inaccessible corridors.

Unmanned aerial vehicles (UAVs) provide a flexible platform for collecting close-range visual data. However, UAV inspection workflows generate large volumes of images that must be examined to identify relevant components and visible defects. Deep learning can automate part of this process by localizing assets and classifying their apparent condition. Edge computing enables image analysis near the camera, reducing the need to transmit every image to a remote server.

This work presents a two-stage inspection pipeline. The first stage uses YOLO11n for power-line asset detection. The second stage uses EfficientNetB0-based classifiers to route selected component crops and assess their condition. YOLO11n is selected as a compact detector variant intended for resource-constrained inference. The main contributions are: (i) a tower-grouped, leakage-aware dataset split; (ii) replacement of the earlier YOLOv8s detector with YOLO11n; (iii) visual analysis of the supplied 30-, 50-, and 100-epoch training results; and (iv) an edge-oriented system architecture with explicit separation between implemented image-processing functions and future autonomous-flight integration.

2. Related Work

The InsPLAD benchmark provides annotated aerial imagery for power-line asset inspection and supports the development of object-detection methods for transmission infrastructure [1]. One-stage detectors in the YOLO family are commonly used where object localization and inference efficiency are important. Ultralytics describes YOLO11 as an updated detection family with a nano variant (YOLO11n) and a feature-extraction backbone, feature-fusion neck, and detection head [2]. The YOLO11 model configuration uses C3k2 blocks and a C2PSA attention module in its backbone, followed by multi-scale feature fusion and detection [3].

EfficientNet introduced a compound scaling strategy for convolutional neural networks and has been used as a feature-extraction backbone in image-classification tasks [4]. In an inspection workflow, a detector can first locate a component, after which a dedicated classifier analyzes the crop. Such a cascade must be evaluated end-to-end because inaccurate or incomplete crops can reduce downstream classification performance.

Edge platforms such as the Raspberry Pi 5 provide compact computing for localized inference and data handling [5]. Camera Module 3 uses the Sony IMX708 sensor and supports autofocus functionality [6]. A FastAPI-based service can expose telemetry and inspection results through web interfaces [7].

3. Proposed Methodology

3.1 System Architecture

The proposed system is organized into an image-acquisition layer, a deep-learning inference layer, a persistence layer, and a monitoring interface. The Arducam Camera Module 3 captures visual frames and transfers them to the Raspberry Pi 5 through the MIPI CSI-2 interface. The inference engine executes the YOLO11n detector and forwards selected regions of interest to the hierarchical EfficientNetB0 classification pipeline. Detection and condition results are stored in SQLite, while a FastAPI application provides live video and inspection telemetry to a ground monitoring interface.

The flight-control and AI-processing functions are separated in the dual-hardware concept. The Raspberry Pi 5 performs image processing and telemetry functions. A Pixhawk flight controller and MAVLink-based closed-loop autonomous flight integration are not physically implemented in the reported system and remain future work.

3.2 Dataset and Data Reconciliation

The detection stage uses the InsPLAD dataset, which contains aerial photographs collected along transmission corridors. The source technical audit reports 10,607 image records and 28,959 bounding-box annotations. After removing 46 duplicate image records and excluding 26 annotations from the non-structural sphere category, the reconciled detection dataset contains 10,561 unique images and 28,933 annotations across 17 retained asset classes.

Partition	Images	Annotations	Tower groups
Training	7,368	19,768	71
Validation	2,122	6,085	21
Testing	1,071	3,080	11
Total	10,561	28,933	103

To reduce visual background leakage, images were grouped by transmission-tower identifier. All images associated with a given tower were assigned to only one partition. The Group-Stratified Tower Split used a fixed random seed of 42 and separated 71 towers for training, 21 for validation, and 11 for testing. This approach is intended to measure generalization to previously unseen tower structures more reliably than a random image-level split.

A class-list reconciliation is required before final submission: the supplied confusion-matrix plots display a sphere label in addition to the retained classes, whereas the source dataset audit states that sphere annotations were excluded and reports 17 classes. The final dataset YAML and confusion-matrix export should be checked to confirm the exact number and order of classes.

3.3 YOLO11n Object Detection

The detection stage is updated from YOLOv8s to YOLO11n. YOLO11n is the nano-scale variant of the YOLO11 detection family and is intended to provide a compact model option for applications with limited computing resources. The model processes an input image and predicts object categories with corresponding bounding-box coordinates. Its architecture follows the YOLO11 design, which includes feature extraction, multi-scale feature fusion, and a detection head [2], [3].

The supplied experimental plots compare training runs of 30, 50, and 100 epochs. The plots include mAP@0.50, mAP@0.50:0.95, training box loss, training classification loss, training distribution-focal loss, and validation box/classification loss. These curves are included as the experimental evidence available for the YOLO11n revision. The graph files do not provide a machine-readable metrics summary or a clearly stated final checkpoint identifier; consequently, exact precision, recall, mAP values, optimizer settings, batch size, image size, learning rate, and training hardware are not asserted here.

The visual comparison suggests that the 100-epoch run continues to improve the mAP curves beyond the 30- and 50-epoch runs, while its loss curves generally decline over training. This trend indicates continued optimization during the longer run; it does not by itself establish that the 100-epoch checkpoint is the best model on an independent test set. Final model selection should be based on validation results and a separately reported held-out test evaluation.

3.4 Hierarchical Fault Classification

The second stage retains the EfficientNetB0 transfer-learning classifiers described in the source manuscript. A five-class asset router directs detected crops to component-specific condition classifiers for the monitored asset groups: glass insulators, lightning-rod suspensions, polymer-insulator upper shackles, vari-grips, and yoke suspensions. The classifiers distinguish the condition categories defined in the supervised dataset. The source technical report records 11,525 verified cropped images in the supervised classification dataset and evaluation on 1,152 held-out test crops.

The source manuscript reports 95.40% multiclass accuracy, 89.42% balanced accuracy, 96.44% binary Normal/Defective accuracy, and 0.9805 ROC-AUC for this classification stage. These values are retained as source-reported classifier results; they are not YOLO11n detector metrics.

3.5 Cascaded Inference and Edge Implementation

At inference time, YOLO11n generates bounding boxes. The pipeline extracts component regions with context padding and passes the crops to the asset router and corresponding condition classifier. This design allows the condition stage to operate on detector-generated regions rather than ideal manually cropped images. The edge platform described in the source implementation uses a Raspberry Pi 5 with 8 GB RAM and an Arducam Camera Module 3. A multi-threaded inference engine decouples frame acquisition from inference and dashboard communication. SQLite in WAL mode stores inspection records; FastAPI, WebSockets, and MJPEG provide monitoring and streaming.

The source paper's measured detector latency of 94.2 ms and processed throughput of 3.2–3.8 FPS were measured for YOLOv8s, not YOLO11n. They are therefore not transferred to the revised detector. YOLO11n latency, memory use, temperature, and end-to-end throughput must be benchmarked on the target Raspberry Pi 5 before these edge-performance claims can be updated.

4. Experimental Results and Discussion

4.1 Training and Validation Curves

Figures 1 and 2 compare the mAP@0.50:0.95 and mAP@0.50 curves for 30-, 50-, and 100-epoch runs. Both metrics rise during training and approach a plateau toward the later epochs. The 100-epoch curve extends beyond the shorter runs and reaches the highest displayed endpoint in the supplied plots. The plots are visual comparisons; exact endpoint values should be taken from the original results.csv or validation report rather than estimated from the raster images.

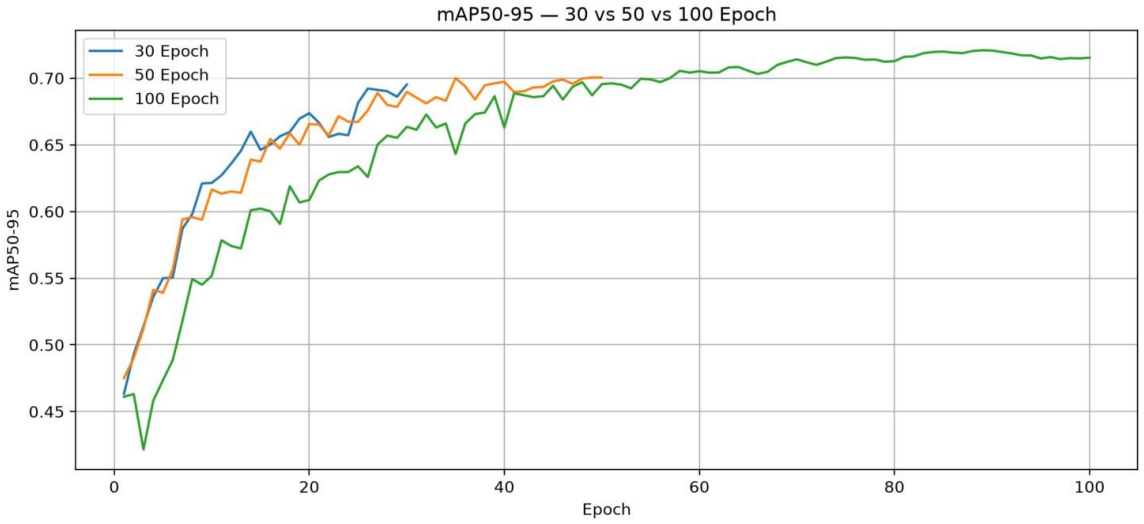


Fig. 1. mAP@0.50:0.95 comparison for 30-, 50-, and 100-epoch YOLO11n training runs.

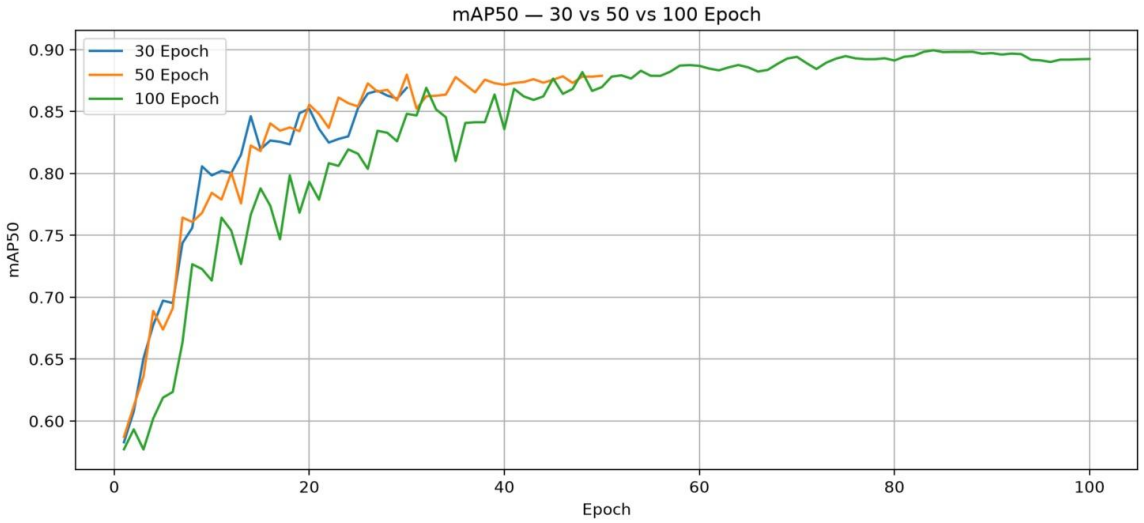


Fig. 2. mAP@0.50 comparison for 30-, 50-, and 100-epoch YOLO11n training runs.

4.2 Training and Validation Loss

The box-regression, classification, and distribution-focal training losses decrease over the training period. The validation box and classification losses also show an overall downward trend, with fluctuations during early epochs. The curves indicate that the model learns progressively from the training data. However, loss reduction alone is not sufficient to establish generalization; the held-out test metrics and class-wise error analysis are required for that purpose.

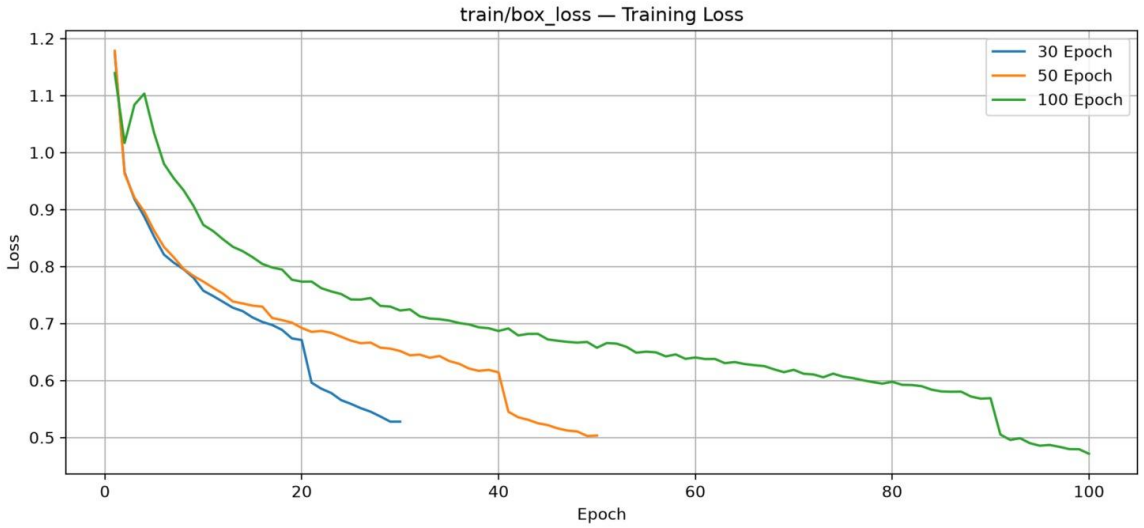


Fig. 3. YOLO11n training box loss for 30-, 50-, and 100-epoch runs.

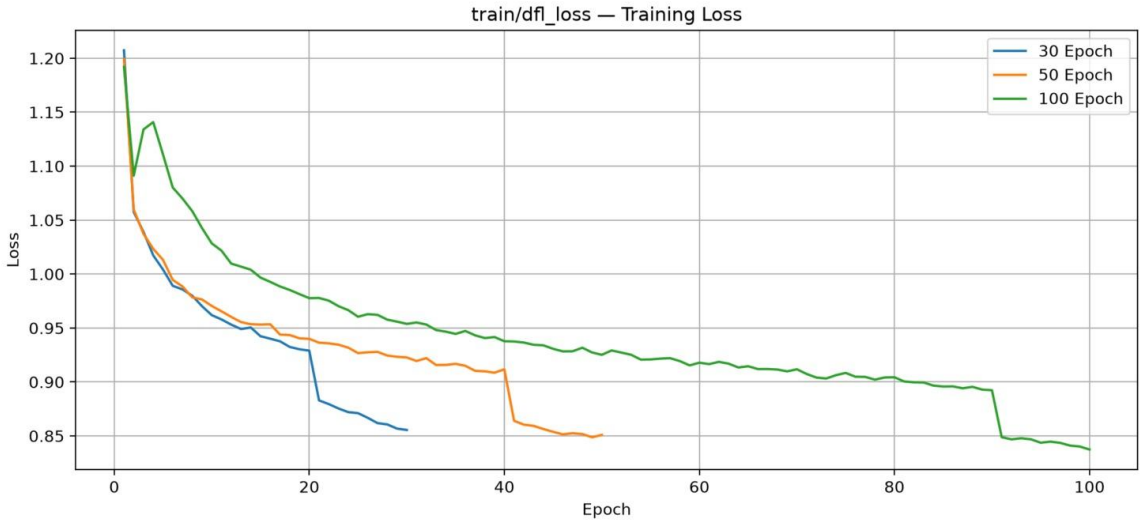


Fig. 4. YOLO11n training distribution-focal loss for 30-, 50-, and 100-epoch runs.

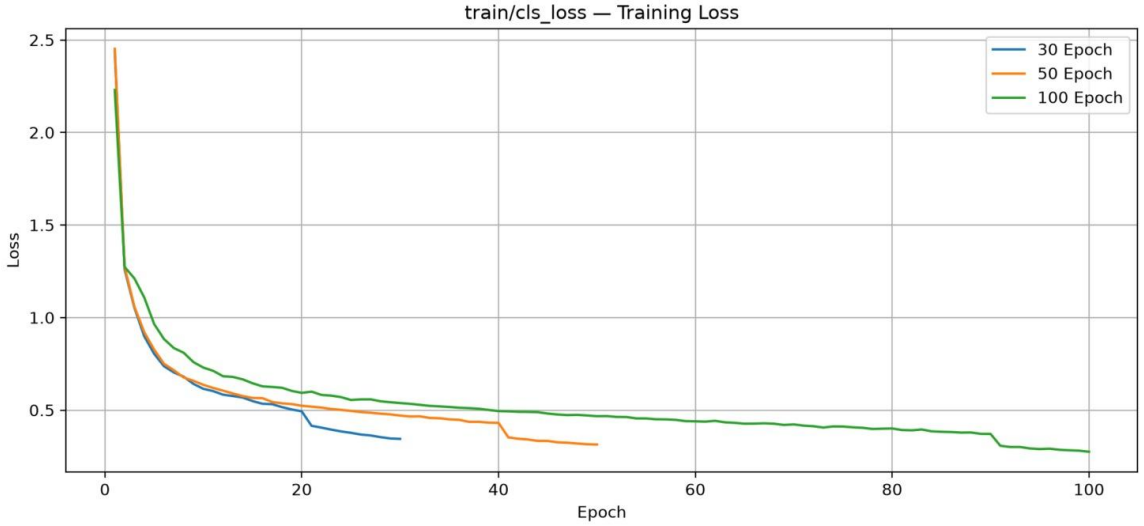


Fig. 5. YOLO11n training classification loss for 30-, 50-, and 100-epoch runs.

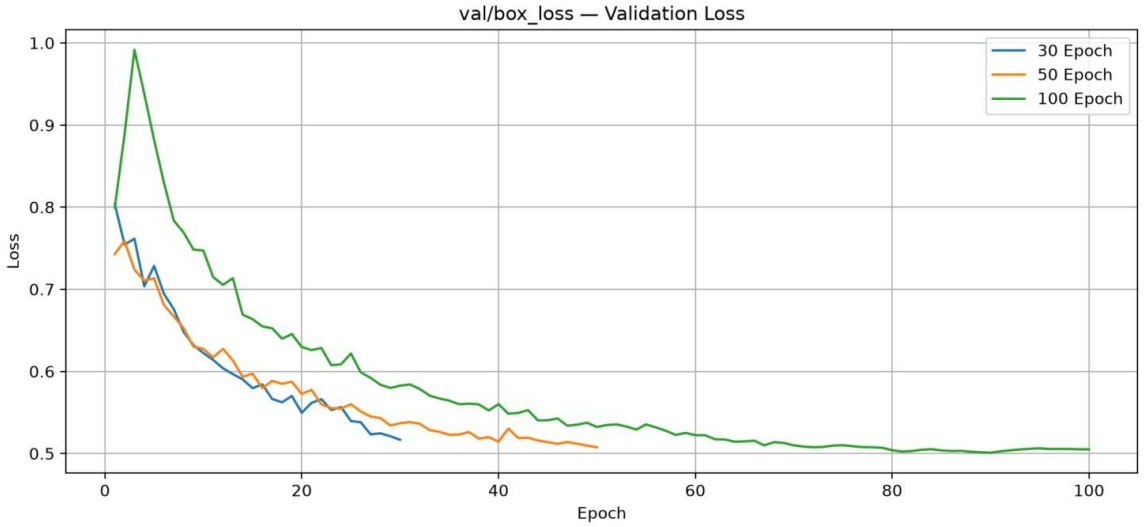


Fig. 6. YOLO11n validation box loss for 30-, 50-, and 100-epoch runs.

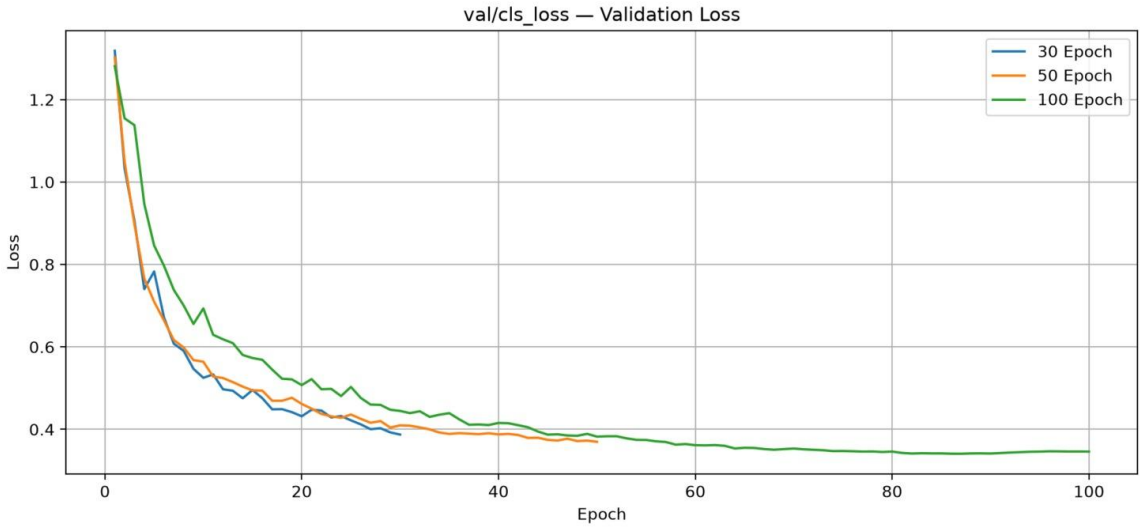


Fig. 7. YOLO11n validation classification loss for 30-, 50-, and 100-epoch runs.

4.3 Confusion-Matrix Analysis

The supplied confusion matrix and normalized confusion matrix provide class-wise evidence for the displayed YOLO11n run. Several categories exhibit strong diagonal concentration, including yoke suspension, stockbridge damper, polymer insulator, glass insulator, tower ID plate, vari-grip, and spiral damper. Lower normalized diagonal values are visible for some smaller or visually similar shackle categories, including glass-insulator big shackle, glass-insulator small shackle, and glass-insulator tower shackle. The background row and column also show missed detections and background-related false detections. This indicates that aggregate performance should be supplemented with class-wise precision, recall, and support.

The confusion-matrix plots do not identify whether they correspond to the 30-, 50-, or 100-epoch checkpoint. They should be labelled with the exact checkpoint and evaluation split before journal submission. The class-label discrepancy described in Section 3.2 must also be resolved.



Fig. 8. Confusion matrix supplied for the YOLO11n detector; checkpoint and evaluation split require confirmation.



Fig. 9. Normalized confusion matrix supplied for the YOLO11n detector; checkpoint and evaluation split require confirmation.

4.4 Condition Classification and System-Level Evaluation

The source-reported EfficientNetB0 classification results are 95.40% multiclass accuracy, 89.42% balanced accuracy, 96.44% binary Normal/Defective accuracy, and 0.9805 ROC-AUC on the stated held-out crop set. These measurements characterize the classifier stage and should be kept distinct from the YOLO11n detector evaluation.

The previously reported cascaded defect precision (88.62%), defect recall (84.24%), and F1-score (86.37%) were generated using the earlier YOLOv8s detector and its detector-generated crops. They are not presented as YOLO11n end-to-end results. A new cascade evaluation is required after integrating the selected YOLO11n checkpoint.

4.5 Edge Performance and Implementation Boundaries

The source implementation reports a Raspberry Pi 5 companion computer, camera acquisition, SQLite persistence, and a FastAPI monitoring dashboard. The previous detector latency, RAM utilization, SoC temperature, and throughput measurements belong to the YOLO implementation and must not be attributed to YOLO11n. A fresh benchmark should report warm-up policy, input resolution, precision mode, batch size, number of measured frames, mean and percentile latency, peak RAM, temperature, and sustained FPS.

GPS parsing was bench-tested through a serial loopback arrangement in the source technical audit. Outdoor autonomous flight testing was not completed. Pixhawk hardware was not connected and MAVLink software was not implemented; autonomous waypoint control, closed-loop visual servoing, and flight-level sag or clearance measurement are therefore not represented as completed experimental results.

5. Conclusion

This revised paper presents a UAV-oriented power-line asset inspection framework in which YOLO11n replaces the earlier YOLOv8s detector, while the EfficientNetB0-based hierarchical condition-classification stage and edge-monitoring architecture are retained. The InsPLAD detection dataset is described using the tower-grouped split reported in the source technical audit. The supplied YOLO11n plots compare 30-, 50-, and 100-epoch runs and show rising mAP curves and generally decreasing training and validation losses. The confusion matrices indicate strong responses for several asset categories but also reveal weaker performance for selected shackle-related classes and background confusions. Since the uploaded plots do not provide an explicit numerical metrics report, this revision does not claim exact YOLO11n precision, recall, mAP, inference latency, or end-to-end defect F1-score. The next validation step is to export the selected checkpoint's results.csv, test metrics, class-wise report, and Raspberry Pi 5 benchmark, then update the quantitative results. Pixhawk/MAVLink integration and outdoor flight validation remain future work.

References

[1] InsPLAD: Insulator and Power Line Asset Dataset for UAV Inspection, IEEE Transactions on Power Delivery, 2020. Complete author, volume, issue, page, and DOI details should be verified against the original publication.

[2] Ultralytics, “YOLO11,” Ultralytics Documentation. https://docs.ultralytics.com/models/yolo11 (accessed Nov. 3, 2026).

[3] Ultralytics, “YOLO11 Model Architecture Configuration,” GitHub repository. https://github.com/ultralytics/ultralytics/blob/main/ultralytics/cfg/models/11/yolo11.yaml (accessed Nov. 3, 2026).

[4] M. Tan and Q. V. Le, “EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks,” in Proc. 36th International Conference on Machine Learning (ICML), 2019.

[5] Raspberry Pi Ltd., “Raspberry Pi 5 Hardware Specifications and Peripherals Guide,” 2023. https://www.raspberrypi.com/documentation/computers/raspberry-pi.html

[6] Sony Semiconductor Solutions, “IMX708 Diagonal 7.4 mm 11.9M-Effective Pixel Color CMOS Image Sensor Specification,” 2023.

[7] S. Ramírez, “FastAPI Framework Documentation,” 2024. https://fastapi.tiangolo.com/

[8] Arducam Technology, “Arducam Camera Module 3 with Autofocus for Raspberry Pi: User Manual,” 2023.