International Journal of Research in Vocational Studies (IJRVOCAS)
Vol. 6, No. 2, August 2026, pp. 39~46
Print ISSN 2777-0168| Online ISSN 2777-0141| DOI prefix: 10.53893
https://journal.gpp.or.id/index.php/ijrvocas/index

Deep Learning and N8N-Based Soybean Defect Recognition and Traceable Quality Grading System

Rafie Hamizan Alhafiz[1, 2], Pola Risma[1, *], Hsien-Wei Tseng[3], Chun-Chieh Fan[2]

[1]Electronics Engineering Department, Politeknik Negeri Sriwijaya, Palembang, Indonesia
[2]Department of Electrical Engineering, Saint John's University, New Taipei City, Taiwan
[3]Department of Electrical and Computer Engineering, Tamkang University, New Taipei City, Taiwan

Email address:
polarisma@polsri.ac.id
[*]Corresponding author : Pola Risma

To cite this article:
Al Hafiz, R. H., Risma, P., Tseng, H.-W., & Fan, C.-C. (2026). Deep Learning and N8N-Based Soybean Defect Recognition and Traceable Quality Grading System. *International Journal of Research in Vocational Studies (IJRVOCAS), 6(2), 39–46. https://doi.org/10.53893/ijrvocas.v6i2.532.*

Received: 05 14, 2026; **Revised**: 07 20, 2026; **Accepted**: 08 13, 2026; **Published**: 08 22, 2026

Abstract: Soybean is an important food commodity whose market value is strongly influenced by seed quality. However, quality inspection in many small and medium agro-industries is still performed manually, making the process slow, subjective, and poorly documented. This study develops an automated soybean inspection system that combines deep-learning-based defect recognition with traceability-supported quality reporting through workflow automation. A YOLOv11n model was trained on a self-collected dataset of 344 annotated images covering five classes, namely Intact, Broken, Skin Damaged, Spotted, and Immature, which was expanded to 1,032 images through augmentation. The model runs on a Raspberry Pi 5 inspection station, where accumulated detection results are converted into a defect rate that determines the quality grade of each batch. An n8n workflow then forwards every inspection result to the Gemini 2.5 Flash large language model to generate a narrative quality-control report, and each batch is stored in an SQLite database that can be accessed through a history viewer to support traceability. Validation results show a precision of 92.73%, a recall of 90.71%, an mAP@0.50 of 96.54%, and an mAP@0.50–0.95 of 88.67%. Functional testing demonstrates that the system produces consistent grades, accumulates detections from multiple captures into a single batch, and generates reports automatically, while still producing a rule-based report when the language model service is unavailable. The proposed system therefore offers an automated, low-cost, and traceability-supported approach to soybean quality inspection that is suitable for deployment on edge devices.

Keywords: Soybean Defect Detection, YOLOv11, Quality Grading, n8n, Large Language Model, Traceability

1. Introduction

Soybean (*Glycine max*) is one of the most important agricultural commodities because it is widely used as a source of plant protein, edible oil, food products, and animal feed [1]. As a traded commodity, seed quality is a primary factor that determines market value, processing yield, and the safety of derived products. Quality inspection is therefore required to identify visual defects on soybean seeds, such as broken seeds, damaged seed coats, spotted surfaces, and immature seeds [2], before the seeds enter the next stage of distribution or processing.

In many small and medium agro-industries, however, soybean quality inspection is still carried out manually through visual observation of the physical appearance of the seeds [3]. Because defects such as broken seeds, damaged seed coats, spots, and immaturity appear as visual features on the seed

surface, the assessment depends heavily on the operator's visual accuracy, so the results tend to be subjective and inconsistent [3]. Manual inspection is also time-consuming because every seed must be examined one by one [4], and it does not produce digital documentation that can be stored as an inspection history. This condition makes traceability and auditing difficult when they are needed at a later time.

Along with technological development, the agro-industrial sector has started to adopt digital systems that store all inspection data in a structured way so that every product batch can be traced back [5]. At the same time, computer vision and deep learning have shown strong performance in automatically detecting various defects in seeds and agricultural products [6], [7]. Several studies have applied deep learning to soybean seed quality assessment, including the fusion of convolutional neural networks with vision transformers [8], hyperspectral imaging combined with attention-based networks [9], computational approaches for quantifying seed defects [10], improved YOLO architectures for surface defect detection [11], and integrated deep-learning-based quality detection systems [12]. Real-time recognition of full-surface soybean defects and large-scale seed classification have also been demonstrated [12], [13]. These studies generally report accuracy above 90%, confirming the reliability of deep learning for recognizing the physical condition of soybean seeds.

Nevertheless, most of the existing research still focuses on the detection or classification stage alone and has not integrated quality grading, automatic report generation, and inspection history storage into a single unified system. The follow-up actions after detection, such as grade determination, automated reporting, and deployment on low-cost edge devices, are rarely addressed.

Based on these problems, this study develops an integrated soybean inspection system that combines YOLOv11-based defect recognition, rule-based quality grading, and traceable automatic reporting on a low-cost edge device. The main contributions of this work are as follows: (1) a YOLOv11n model trained on a self-collected dataset to detect and classify soybean seeds into five categories, namely Intact, Broken, Skin Damaged, Spotted, and Immature; (2) a rule-based quality grading method that converts accumulated detection results into a defect rate and a batch grade; (3) an n8n workflow integrated with the Gemini 2.5 Flash large language model to generate automatic quality-control reports with a rule-based fallback mechanism; and (4) a traceability system based on SQLite with a history viewer that stores and redisplays every inspection record. The remainder of this paper describes the materials and methods, presents the experimental results and discussion, and closes with the conclusions.

2. Materials and Methods

2.1. System Overview

The system is designed as a low-cost tabletop inspection station with a Raspberry Pi 5 as the main computing device. All detection processes run directly on the edge device, so the system can still be used in locations with limited connectivity, such as warehouses or soybean collection points. The system consists of a camera for image acquisition, a Raspberry Pi 5 that runs YOLOv11n inference, an n8n workflow with the Gemini 2.5 Flash large language model for automated reporting, an SQLite database for storing inspection results, and a graphical interface for the operator. The overall workflow of the system is shown in Figure 1.

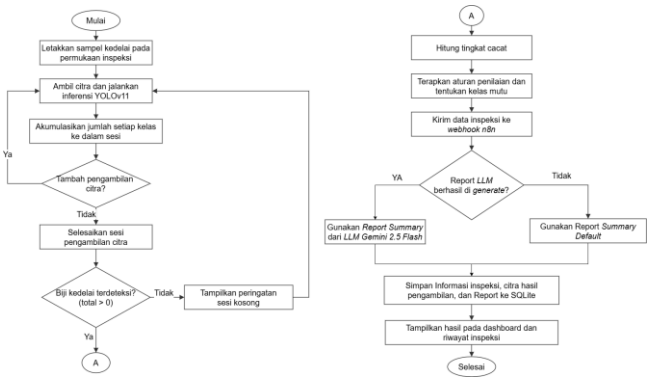

Figure 1. *Overall workflow of the soybean inspection system.*

In operation, one batch supports a multi-capture session, in which a single inspection session can consist of several image captures whose detection results are accumulated. In this way, the samples in one batch do not need to be placed at once within a single camera field of view, so the number of inspected samples is not limited by the inspection area. When the session is finished, the system verifies that at least one seed has been detected, computes the defect rate, determines the grade, requests a narrative report through n8n, stores the result in the database, and displays the result on the dashboard and the history viewer.

2.2. Inspection Station Hardware

The mechanical structure is kept simple so that the research focuses on the inspection function. The camera is mounted facing downward at a fixed distance from the sample area, and the soybean seeds are placed on a black background with consistent lighting provided by a ring light. The controlled acquisition condition increases the contrast between the seeds and the background and keeps the operational images consistent with the training data.

A Raspberry Pi 5 with 4 GB of RAM is used as the main computing unit for image acquisition, model inference, grade determination, interface operation, and data storage. Low-power single-board computers have been shown to be adequate for lightweight deep learning inference in visual inspection applications [14], [15]. A Raspberry Pi Camera Module 3 is connected through the CSI interface and operated at a resolution of 1920 × 1080 pixels as a compromise between image detail and computational load. The system is powered by a 5 V/5 A power supply according to the official Raspberry Pi 5 specification, while the ring light is supplied directly from the 5 V and GND pins of the Raspberry Pi and stays on during inspection to keep the illumination even.

2.3. Defect Classes and Grading Rules

The system distinguishes soybean seeds into five classes, consisting of one non-defect class and four defect classes. These classes are used because they have visual characteristics that can be observed in images and correspond to the inspection objectives of this study. The definition of each class is presented in Table 1.

Table 1. *Definition of the soybean seed classes.*

Class	Type	Description
Intact	Non-defect	Whole seed without visible damage, with a relatively smooth surface and uniform color.
Broken	Defect	Seed that is broken, split, clearly cracked, or present only as a partial structure.
Skin Damaged	Defect	Outer seed coat is peeled, scratched, or torn, while the main body of the seed is still whole.
Spotted	Defect	Seed surface shows local discoloration or dark spots.
Immature	Defect	Seed that is not fully mature and generally greenish in color.

The detection results for each class are converted into a batch-level quality conclusion. The defect rate is computed using (1), where the total number of defective seeds is the sum of the detections of the Broken, Skin Damaged, Spotted, and Immature classes.

$$defect\ rate = (N_{defect} / N_{total}) \times 100\% \quad (1)$$

The defect rate is then mapped into four grade levels according to the rules in Table 2. The thresholds of 5%, 10%, and 20% are set as initial operational rules to differentiate quality levels, giving a defect tolerance that doubles at each grade step, while values above 20% are categorized as Reject so that the system remains sensitive to a high number of defects. These thresholds are not claimed as an official quality standard and can be adjusted based on validation results or user needs. The grading rules are defined in a single central function as a single source of truth and are used by the interface, the storage process, and the report generation, preventing inconsistent grade decisions between components.

Table 2. *Grading rules based on the defect rate.*

Grade	Defect Rate Threshold
Grade A	≤ 5%
Grade B	> 5% and ≤ 10%
Grade C	> 10% and ≤ 20%
Reject	> 20%

2.4. Dataset Collection, Preprocessing, and Augmentation

The dataset was collected independently using the Raspberry Pi 5 and the camera module. Samples were placed on a black background with consistent lighting, and the acquisition condition was made identical to the operational condition of the prototype to reduce differences in image characteristics between modelling and deployment. Sample images are shown in Figure 2. Each image was uploaded to Roboflow and manually annotated with bounding boxes for the five classes.

Figure 2. *Samples of the collected soybean seed images.*

The collected dataset consists of 344 images divided into training, validation, and test sets with a ratio of 80:10:10. The split was performed before augmentation so that augmented images derived from one source image belong to only one subset, avoiding data leakage between the sets. During preprocessing, the image orientation was adjusted automatically based on EXIF metadata. Augmentation produced three outputs for every source image using random horizontal and vertical flips, 90° rotation, random rotation of ±15°, brightness changes of ±15%, and Gaussian blur of up to 0.8 pixels, following common augmentation practice for deep learning [16]. The augmentation types were chosen to resemble real conditions that may occur during inspection. The resulting data distribution is presented in Table 3.

Table 3. *Dataset distribution before and after augmentation.*

Stage	Train	Validation	Test	Total
RAW dataset	275	34	35	344
Split ratio	80%	10%	10%	100%
After augmentation	825	102	105	1,032

2.5. YOLOv11 Model Training and Deployment

The detection model used in this study is the nano variant of YOLOv11 (YOLOv11n). YOLO is a one-stage object detection approach that predicts bounding boxes and class labels in a single forward pass [17], and YOLOv11 introduces architectural enhancements that improve accuracy and efficiency [18]. The nano variant was chosen because of its compact model size and relatively low computational requirement, making it suitable for the Raspberry Pi 5 without a GPU accelerator. The YOLOv11 architecture is shown in Figure 3. The input image is processed by the backbone to extract visual features at several resolution levels, the neck combines features from different scales, and the head produces predictions in the form of bounding boxes, class labels, and confidence scores.

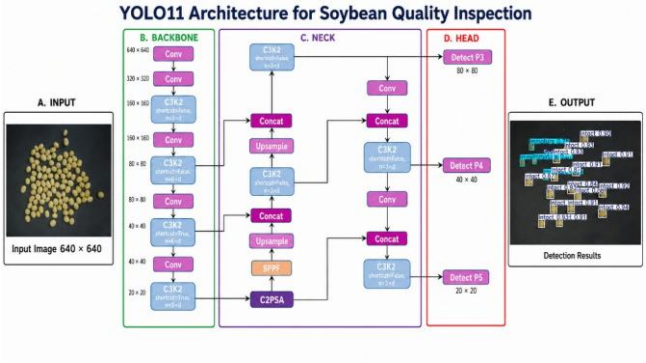

Figure 3. *YOLOv11 architecture.*

Modelling was performed on Google Colab with GPU support, while Google Drive was used to store the weights and modelling results. The model was trained for 100 epochs, and the best weights (best.pt) were deployed to the Raspberry Pi 5 for inference in an isolated Python virtual environment. The initial confidence threshold was set at 0.60 and later validated using the F1–confidence curve during evaluation. Model performance was evaluated on the validation set using precision, recall, mAP@0.50, and mAP@0.50–0.95, complemented by the precision–recall curve, the normalized confusion matrix, the training curves, and the F1–confidence curve.

2.6. Workflow Automation, Report Generation, and Data Storage

The automatic report is generated through an n8n workflow that runs locally on the Raspberry Pi. n8n is an open-source workflow automation platform that supports integration with artificial intelligence services [19]. The workflow consists of a Webhook node that receives the inspection data in JSON format, an AI Analysis node that composes the quality-control narrative, and a Respond to Webhook node that returns the narrative to the application. The integration structure is shown in Figure 4. In the AI Analysis node, Gemini 2.5 Flash [20] is accessed through the Google AI Studio API. The model receives a prompt containing the structured inspection data and produces a narrative about the batch condition, the dominant defect type, the grade result, and handling suggestions.

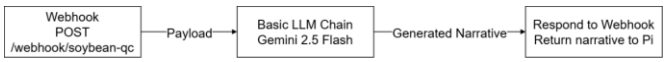

Figure 4. *Block diagram of the n8n and Gemini 2.5 Flash integration.*

The application sends the payload to the n8n webhook and waits for a response for at most 30 seconds. If n8n or Gemini cannot be reached, times out, or returns an invalid format, the application uses a fallback narrative generated automatically by the system. This mechanism keeps the inspection process running when the external service is disrupted.

All inspection results are stored in an SQLite database, a lightweight embedded database suitable for ARM-based platforms [21], so that every batch can be traced back. The database consists of two related tables: the inspections table stores one record per batch, including the inspection time, the number of detections per class, the total number of seeds, the total number of defects, the defect rate, the grade, the report file location, and the narrative; the inspection_images table stores the list of annotated images from each capture, with batch_id as a foreign key. Through this one-to-many relation, one batch can have more than one annotated image. The database schema is shown in Figure 5. Each batch is automatically numbered in the format B[YYYYMMDD]-[sequence number], which keeps batch numbers unique and chronologically sortable. The user interface was developed with the CustomTkinter library and consists of an inspection dashboard and a history viewer.

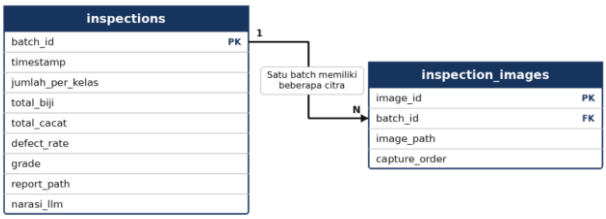

Figure 5. *SQLite database schema.*

3. Results and Discussion

3.1. System Realization

The physical system consists of a Raspberry Pi 5, a Camera Module 3 mounted facing downward, a black base as the inspection area, and a white LED lamp as the light source. The camera and lamp are mounted on a tripod-shaped holder so that their positions and distances to the inspection area remain the same in every session. The realized inspection station is shown in Figure 6. The black base provides good contrast against the yellowish-brown color of the soybean seeds, which helps the model distinguish the seeds from the background during inference, while the LED lamp above the inspection area provides even illumination and minimizes shadows. The Raspberry Pi 5 runs the entire computation, and the system is operated from a computer through VNC because the Raspberry Pi is used without a dedicated monitor.

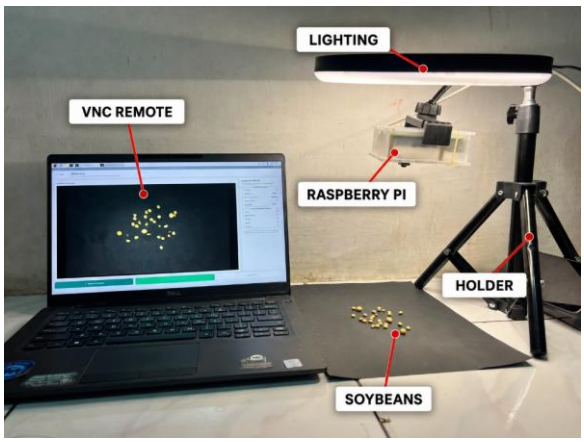

Figure 6. *Realized tabletop soybean inspection station.*

3.2. Model Evaluation

The model achieved a precision of 0.9273, a recall of 0.9071, an mAP@0.50 of 0.9654, and an mAP@0.50–0.95 of 0.8867 on the validation set, as summarized in Table 4. The precision above 0.92 indicates that most of the detections produced by the model are correct, while the recall of 0.9071 shows that more than 90% of the objects in the validation data were found by the model. The relatively small gap between mAP@0.50 and mAP@0.50–0.95 also indicates that the predicted bounding boxes are precisely located with respect to the actual objects.

Table 4. *Overall validation metrics of the YOLOv11n model.*

Metric	Value
Precision	0.9273
Recall	0.9071
mAP@0.50	0.9654
mAP@0.50–0.95	0.8867

The per-class evaluation in Table 5 shows that the Immature, Intact, and Spotted classes obtained mAP@0.50 values above 0.97. These classes have distinctive visual features: Immature seeds are greenish, Intact seeds have a smooth surface, and Spotted seeds have contrasting spots. The Broken class obtained 0.960 because the shape of broken seeds is more varied. The Skin Damaged class has the lowest performance, with an mAP@0.50 of 0.920 and an mAP@0.50–0.95 of 0.849, because the seed coat damage is relatively thin and visually similar to the Intact class, especially when the damage is only a fine crack on the coat surface.

Table 5. *Validation metrics for each class.*

No	Class	mAP@0.50	mAP@0.50–0.95
1	Intact	0.982	0.948
2	Immature	0.994	0.919
3	Broken	0.960	0.854
4	Skin Damaged	0.920	0.849
5	Spotted	0.971	0.865
	All classes	0.965	0.887

The precision–recall curves for each class are shown in Figure 7. The curves of the Immature, Intact, Spotted, and Broken classes are relatively flat at the top of the graph, which means precision is maintained across almost the entire recall range. The Skin Damaged class shows a faster precision drop as recall increases, indicating that when the model is forced to find more Skin Damaged objects, the number of incorrect detections also increases.

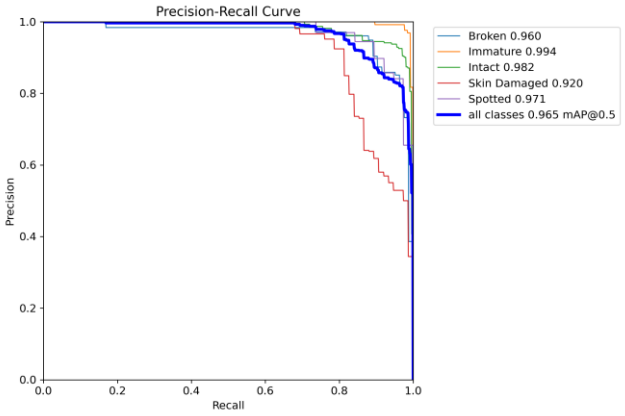

Figure 7. *Precision–recall curves of the YOLOv11n model.*

The normalized confusion matrix on the validation set is shown in Figure 8. Most predictions lie on the main diagonal, indicating that the model classifies each class correctly in the majority of cases. The most frequent errors occur in the Skin Damaged class, where some objects are predicted as Intact or are not detected at all and fall into the background column. Misclassification toward Intact shows that subtle coat damage is still difficult to distinguish from a normal seed surface, while objects that fall into the background are generally seeds with very thin damage whose detection confidence is below the threshold. From a system-usage perspective, this error pattern deserves attention because Skin Damaged seeds predicted as Intact can make the computed defect rate slightly lower than the actual condition.

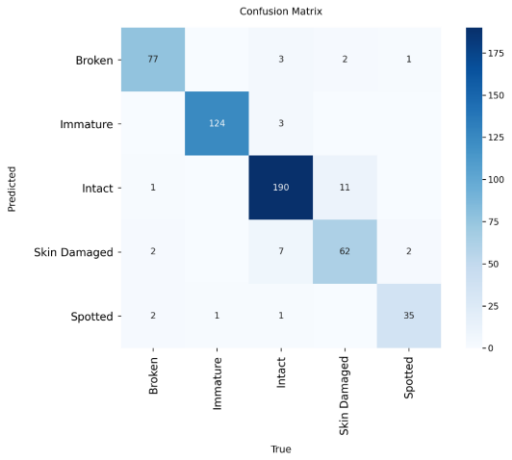

Figure 8. *Normalized confusion matrix on the validation set.*

The training curves over 100 epochs show that the box, classification, and distribution focal losses on both the training and validation data drop sharply in the early epochs and then flatten until the end of training, moving in the same direction without the validation loss turning upward, which indicates convergence without significant overfitting. The metric curves stabilize from around epoch 50, with mAP@0.50 staying above 0.96, so 100 epochs are considered sufficient.

The F1–confidence curve in Figure 9 shows that the highest F1 score of 0.92 is obtained at a confidence level of 0.608, and the curve is relatively flat around its peak. The confidence threshold of 0.60 used in the system implementation is very

close to this optimal point, so low-confidence detections are filtered out without sacrificing the number of detected objects, and the defect rate computation remains reliable.

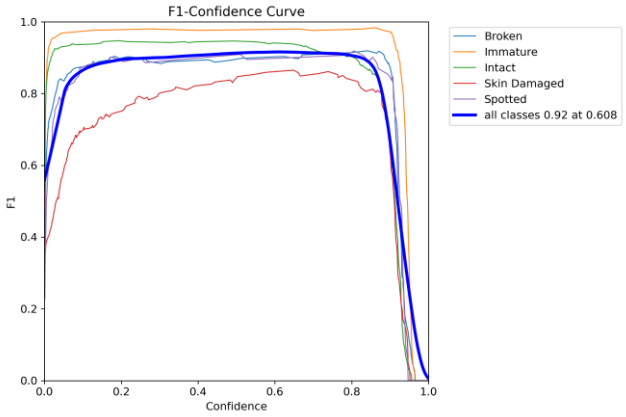

Figure 9. *F1–confidence curve of the YOLOv11n model.*

3.3. Inference Testing

Inference testing was conducted in two scenarios, namely single-class samples and mixed samples. In the single-class tests, the model recognized all five classes well, with confidence scores in the range of 0.82–0.99 for Intact, 0.88–0.95 for Immature, 0.88–0.94 for Spotted, 0.87–0.95 for Broken, and 0.88–0.95 for Skin Damaged. Under simple and controlled image conditions, the detections are stable because the objects do not overlap and each class has sufficiently clear visual characteristics.

In the mixed-sample test shown in Figure 10, the model detected the Intact, Immature, Spotted, and Skin Damaged classes simultaneously with confidence values of approximately 0.64–0.94. The decrease in confidence on some detections is explained by the increased image complexity, where objects are closer together and shape similarity between classes makes feature separation less optimal. A few objects were also missed in dense layouts. Nevertheless, most classes were still correctly recognized, so the model performs consistently on simple images and remains adequate under more complex multi-object conditions.

Figure 10. *Detection results on a mixed multi-class sample.*

3.4. End-to-End System Testing and Quality Grading

The inspection interface is the main application view used by the operator. The display is divided into a camera preview on the left and an inspection result panel on the right, with buttons for capturing and detecting, finishing the session and grading the batch, and resetting the session. The interface during an end-to-end test is shown in Figure 11. In the first capture, the system detected 102 seeds consisting of 98 Intact and 4 Skin Damaged seeds, giving a temporary defect rate of 3.92%.

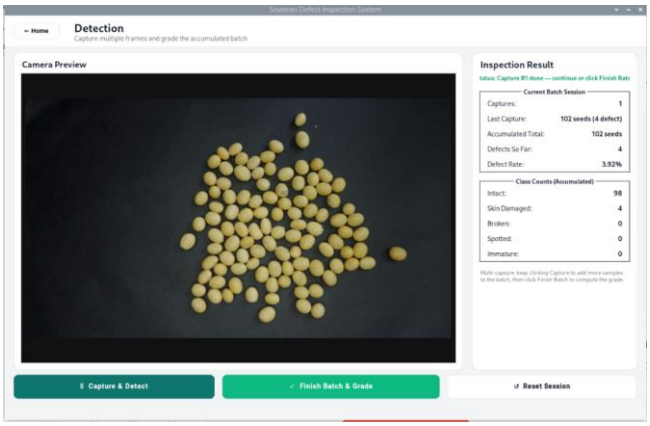

Figure 11. *Inspection interface during an end-to-end test.*

After the session was completed, the batch was stored with the identity B20260701-008 and consisted of two captures. The final result contained 128 seeds, namely 119 Intact, 4 Skin Damaged, 0 Broken, 2 Spotted, and 3 Immature. The system counted 9 defective seeds, computed a defect rate of 7.03%, and assigned Grade B. The second capture added 26 seeds to the session, and the stored totals matched the sum of both captures, showing that the multi-capture accumulation works correctly and is used as the basis for the defect rate computation and grade determination. Grade determination and report generation run in a background process so that the interface remains responsive.

Grading consistency was verified by comparing the stored defect rates and grades of several batches in the SQLite database. Batch B20260701-007 contained 15 seeds without defects and received Grade A; batch B20260701-008 contained 128 seeds with 9 defects (7.03%) and received Grade B; batch B20260701-009 contained 25 seeds with 4 defects (16.00%) and received Grade C; and batch B20260701-004 contained 21 seeds with 5 defects (23.81%) and was declared Reject. These results show that the system applies the grading rules consistently and that all inspection results are recorded in the database.

3.5. Automated Reporting, Fallback Mechanism, and Traceability

The n8n workflow runs as a self-hosted service on the Raspberry Pi 5 and consists of Webhook, AI Analysis, Google Gemini Chat Model, Output Parser, and Respond to Webhook nodes. During testing, the workflow was active and successfully processed a request in 5.153 seconds. The application sent the inspection data to n8n, received the report generated by Gemini 2.5 Flash, displayed it in the result window, and stored it together with the batch data.

The fallback mechanism was tested by deactivating the n8n workflow before the inspection was completed. Because no valid response was received, the application automatically generated a local rule-based report without stopping the process. The test result was stored as batch B20260701-009 with 25 seeds, 4 defective seeds, a defect rate of 16.00%, and Grade C, with Immature as the dominant defect class. Although the language model could not be accessed, the system still displayed the report, stored the batch data, and completed the entire inspection process. The fallback report is shorter than the report from Gemini 2.5 Flash but still contains the main information required for documentation and traceability.

The history viewer displays the inspection history with a batch list on the left and inspection details on the right, as shown in Figure 12. The list contains the batch ID, inspection time, number of captures, total seeds, defect rate, and grade, and is equipped with search and sorting features. When a batch is selected, the system displays the annotated images, the accumulated counts of each class, the computation summary, and the related report; batches with more than one image can be browsed image by image. Record deletion through the history viewer removes the main record, the related image data, and the stored files consistently, supported by the foreign key relation and a cascade delete mechanism.

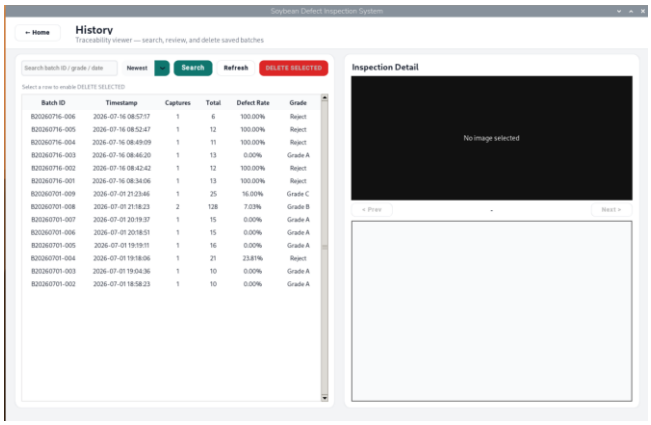

Figure 12. *History viewer for batch traceability.*

4. Conclusion

This study developed an automated soybean inspection system that integrates YOLOv11-based defect recognition, rule-based quality grading, automatic report generation, and traceability on a low-cost edge device. The YOLOv11n model successfully detects and classifies soybean seeds into five categories, achieving a precision of 92.73%, a recall of 90.71%, an mAP@0.50 of 96.54%, and an mAP@0.50–0.95 of 88.67% on the validation set, with the Skin Damaged class as the most challenging because of its visual similarity to Intact seeds. The defect rate computed from the accumulated detections provides a consistent basis for the four-level grading rules, as verified by the stored batches, and the multi-capture mechanism accumulates several captures into a single batch correctly. The integration of n8n and Gemini 2.5 Flash generates narrative quality-control reports automatically, and the fallback mechanism keeps the system functional when the language model service is unavailable. All inspection results are stored in an SQLite database and can be redisplayed through the history viewer, so previous inspections can be traced and verified when needed.

Future work includes expanding the dataset with more varied lighting conditions, seed positions, and defect severities, especially for the Skin Damaged class; adopting multi-view image acquisition so that defects on the hidden side of the seeds can be detected; optimizing inference on the Raspberry Pi 5 through quantization, pruning, or ONNX Runtime; aligning the grading thresholds with official soybean quality standards; and extending the system with a physical sorting mechanism driven by the detection results.

Acknowledgements

The authors would like to thank the Electrical Engineering Department of Politeknik Negeri Sriwijaya, Saint John's University, and Tamkang University for the support and facilities provided during this research.

References

[1] A. Vargas-Almendra, R. Ruiz-Medrano, L. A. Núñez-Muñoz, J. A. Ramírez-Pool, B. Calderón-Pérez, and B. Xoconostle-Cázares, "Advances in Soybean Genetic Improvement," Nov. 01, 2024, *Multidisciplinary Digital Publishing Institute (MDPI)*. doi: 10.3390/plants13213073.

[2] W. Lin *et al.*, "Soybean image dataset for classification," *Data Brief*, vol. 48, 2023, doi: 10.17632/v6vzvfszj6.6.

[3] C. Song *et al.*, "Maize seed appearance quality assessment based on improved Inception-ResNet," *Front. Plant Sci.*, vol. 14, 2023, doi: 10.3389/fpls.2023.1249989.

[4] L. Fan *et al.*, "An annotated grain kernel image database for visual quality inspection," *Sci. Data*, vol. 10, no. 1, Dec. 2023, doi: 10.1038/s41597-023-02660-8.

[5] S. Charlebois, N. Latif, I. Ilahi, B. Sarker, J. Music, and J. Vezeau, "Digital Traceability in Agri-Food Supply Chains: A Comparative Analysis of OECD Member Countries," Apr. 01, 2024, *Multidisciplinary Digital Publishing Institute (MDPI)*. doi: 10.3390/foods13071075.

[6] H. Li, Z. Li, D. Chen, W. Wu, X. He, and H. Mu, "Oak-YOLO: A high-performance detection model for automated Oak seed defect identification," *PLoS One*, vol. 20, no. 8 August, Aug. 2025, doi: 10.1371/journal.pone.0327371.

[7] Y. Che, H. Bai, L. Sun, Y. Fang, X. Guo, and S. Yin, "Real-Time Detection of Varieties and Defects in Moving Corn Seeds Based on YOLO-SBWL," *Agriculture (Switzerland)*, vol. 15, no. 7, Apr. 2025, doi: 10.3390/agriculture15070685.

[8] G. H. Hussien, A. K. Z. Alsaedi, and A. H. M. Dreheeb, "Soybean Seed Classification with Fusion of Convolutional Neural Networks and ViT Technique," *Global Journal of Engineering and Technology Research*, vol. 02, no. 02, Feb. 2026, doi: 10.65150/EP-gjetr/V2E2/2026-03.

[9] J. Gui, H. Xu, and J. Fei, "Non-Destructive Detection of Soybean Pest Based on Hyperspectral Image and Attention-ResNet Meta-Learning Model," *Sensors*, vol. 23, no. 2, Jan. 2023, doi: 10.3390/s23020678.

[10] A. Sable, P. Singh, A. Kaur, M. Driss, and W. Boulila, "Quantifying Soybean Defects: A Computational Approach to Seed Classification Using Deep Learning Techniques,"

Agronomy, vol. 14, no. 6, Jun. 2024, doi: 10.3390/agronomy14061098.

[11] C. Liu *et al.*, "Detection of surface defects in soybean seeds based on improved Yolov9," *Sci. Rep.*, vol. 15, no. 1, Dec. 2025, doi: 10.1038/s41598-025-92429-3.

[12] B. Ji *et al.*, "Automated soybean quality detection system using deep learning," *Smart Agricultural Technology*, vol. 12, Dec. 2025, doi: 10.1016/j.atech.2025.101473.

[13] G. Zhao *et al.*, "Real-time recognition system of soybean seed full-surface defects based on deep learning," *Comput. Electron. Agric.*, vol. 187, p. 106230, 2021, doi: 10.1016/j.compag.2021.106230.

[14] M. Loni, A. Simonsen, M. Alibeigi, S. Daneshtalab, M. Sjodin, and J. Ekman, "A comprehensive survey of deep learning-based lightweight object detection models for edge devices," *Artif. Intell. Rev.*, vol. 57, no. 9, p. 240, 2024, doi: 10.1007/s10462-024-10877-1.

[15] A. Catania and D. Guastella, "Edge AI for industrial visual inspection: YOLOv8-based visual conformity detection using Raspberry Pi," *Algorithms*, vol. 18, no. 8, p. 510, 2025, doi: 10.3390/a18080510.

[16] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *J. Big Data*, vol. 6, no. 1, p. 60, 2019, doi: 10.1186/s40537-019-0197-0.

[17] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, 2016, pp. 779–788. doi: 10.1109/CVPR.2016.91.

[18] R. Khanam and M. Hussain, "YOLOv11: An overview of the key architectural enhancements," *arXiv preprint arXiv:2410.17725*, 2024, [Online]. Available: https://arxiv.org/abs/2410.17725

[19] P. Venkiteela, "n8n: An open-source workflow automation platform for enterprise integration and AI-driven orchestration," *Int. J. Comput. Appl.*, vol. 187, no. 63, 2025, doi: 10.5120/ijca2025926031.

[20] G. DeepMind, "Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities," Google DeepMind, 2025. [Online]. Available: https://storage.googleapis.com/deepmind-media/gemini/gemini_v2_5_report.pdf

[21] Y. Wan, "Embedded database SQLite application based on ARM platform," in *Proceedings of SPIE 13259, International Conference on Automation Control, Algorithm, and Intelligent Bionics (ACAIB 2024)*, 2024, p. 132592U. doi: 10.1117/12.3039708.