INTEL-UNNATI AI&ML JOURNAL
Number 2 (2025), pp. 1–8

Code:IUIT-2025

MACHINE LEARNING

A ROS 2-based System for AI-driven Validation and Traceability of Product Labels with Digital Twin Simulation

Shahazad Abdulla and Hridya Mariam Reji

Saintgits Group of Institutions, Kottayam, Kerala

Abstract: In contemporary high-stakes production, maintaining product integrity and total traceability is an essential problem. Current systems do not have the built-in intelligence to carry out end-to-end, real-time quality checks on each unit. This paper describes an end-to-end, automated system for the validation of product labels, designed entirely within the Robot Operating System 2 (ROS 2) framework. Our approach mimics a factory inspection station where products pass through a multistage AI-based validation pipeline. This pipeline includes RoHS conformity checks, OpenCV-based image quality pre-filtering, QR code content validation with Pyzbar, OCR text validation with EasyOCR, and a new label print quality assessment using a custom-trained TensorFlow/Keras model, which achieved 90 percentage accuracy on our test set. The entire process is visualized in a PyBullet "digital twin" simulation, managed by a Tkinter-based user interface, and all validation outcomes are stored in a persistent SQLite database for full traceability. The event-driven modular structure demonstrates a high-performance, scalable solution for industrial automation today.

Keywords: ROS 2, Industrial Automation, Traceability, Computer Vision, Machine Learning, Digital Twin

1 Introduction

For today's global manufacturing environment in the electronics, motor, and pharmaceutical industries, quality control as well as end-to-end product traceability is a priority. Being able to track a product from the original components through to the end user is essential for

safety recall management, staying compliant with regulations, and protecting brand reputation. However, most current industrial processes continue to rely on isolated or semi-automatic systems for label checking, which is slow, error-prone, and unable to provide a complete quality assurance snapshot for each unit. This creates a huge gap through which products with faulty labels or quality defects can enter the supply chain.

The motivation for this project was to close this gap through the development of a modern, intelligent, and comprehensive solution. We wanted to look beyond simple barcode scanning to a full-scale, multi-modal validation pipeline that could check various aspects of a product's label and quality. In order to obtain the desired modularity and scalability for an industrial application, we designed our system based on the Robot Operating System 2 (ROS 2), a de facto standard in contemporary robotics. This paper describes a Smart Product Labeling and Traceability System utilizing a "digital twin" (DT) for process visualization. The system applies a series of Artificial Intelligence (AI) and Computer Vision (CV) algorithms to execute its validation tasks. Some of the main components are Optical Character Recognition (OCR) for reading text, QR code decoding to check unique identifiers, and a Machine Learning (ML) model trained specifically for assessing the physical print quality of the label. All the validation processes are monitored using a User Interface (UI) and recorded in a permanent database, fulfilling the stringent traceability requirement. The paper demonstrates an overall robust and scalable approach to automated quality control.

2 Related work

Several pieces of work have explored AI in industrial quality control. Raisul Islam et al.'s piece provides an elaborate discussion of deep learning and computer vision techniques used in manufacturing quality control. Their paper discusses the application of CNNs and YOLO models in defect detection and segmentation and provides an overview of the trend and challenges in the field [1].

In robotics, while ROS-based automation systems are common [2], vision is typically only utilized in scanning large-scale infrastructure and not for a multistage quality verification pipeline on a factory floor, as in this research. For instance, Sanchez-Cubillo et al.'s system utilizes autonomous mobile robots and AI to scan critical assets with a focus on navigation and traceability of the data for repair.

The novelty of our project comes from the fact that it is an end-to-end, full system. The papers discussed may include a literature review, a binary part-inspection system, or a mobile robot maintenance method, but our system is new in terms of approach. We have a multi-stage validation pipeline integrating data conformity checks, QR/OCR checks, and a custom ML quality model, all within one ROS 2 architecture. By combining this with a visual digital twin and a persistent traceability log, our work provides a more comprehensive and production-ready model for a smart industrial inspection station.

While existing research provides important building blocks for our project, it often addresses single aspects of a complete industrial automation solution. For example, the review by Saberironaghi et al. [3] looks at deep learning for defect detection. The work by Howard et al. [4] presents an efficient model architecture using MobileNets. On another note, Erős et al. [5] focus on the communication framework with ROS2, while Ammendrup and Barcos [6] highlight the importance of traceability.

Our project's main contribution is the practical combination of these different elements. We are using an efficient MobileNet-based vision model within a ROS2 communication structure to create a comprehensive defect detection and traceability system. This goes beyond theoretical reviews or isolated components and delivers a fully integrated, practical application that connects real-time visual inspection directly to a product's lifecycle log.

3 Methodology

Our system is implemented as a distributed network of communicating nodes using ROS 2 Humble. This kind of modular design, as shown in the computation graph in Fig. 1, allows for clean separation of concerns and scalability. The most important building blocks are a central conductor_node for process orchestration, a pybullet_visualizer_node for visualization of the digital twin, and an ai_validation_node that performs the quality verification.

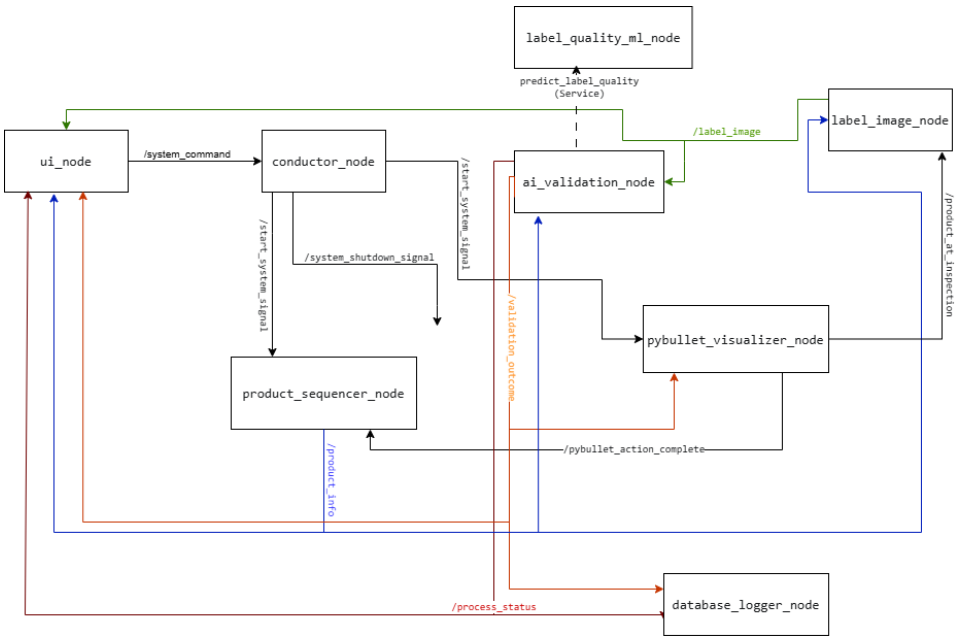

Figure 1: ROS 2 System Architecture showing key nodes and topic communication.

The validation pipeline is executed for each product individually. When provided with a "START" command from the ui_node, the product_sequencer_node publishes data of the first product. The pybullet_visualizer_node positions the product at an inspection location and signals its arrival. This triggers the label_image_node to publish a simulated camera view of the product's label. This is the input to the ai_validation_node, which runs the following tests in sequence:
1. RoHS Compliance Check: Checks the RoHS_Compliant field of the master data of the product.
2. Image Quality PreCheck: Performs an OpenCV-based Laplacian variance calculation to

discard images too fuzzy for safe AI processing.
3. ML Print Quality Check: Calls the label_quality_ml_node service, which uses a custom-trained TensorFlow/Keras model to classify the label's print as "GOOD" or "BAD".
4. QR Code Validation: Uses pyzbar to scan the QR code and checks the content of the QR code against the target serial number.
5. OCR Text Validation: Uses easyocr to read all text present on the label. A custom parsing function subsequently reads and cleanses the Batch ID and Serial Number, which are validated against expected values.

In case a check fails, the process is terminated, and a "REJECTED" status is returned. In case all checks pass, an "ACCEPTED" status is returned. The ultimate decision is forwarded to the pybullet_visualizer_node, which initiates the right accept or reject animation. The entire process is then initiated for the next product.

3.1 Hardware/Software Mapping & Conceptual Embedded System

While the system is demonstrated in a simulation, it is designed for deployment on a physical embedded system. Figure 2 shows the conceptual hardware architecture. A Raspberry Pi 4 would run the high-level ROS 2 nodes (Conductor, UI, and AI Validation), processing camera images and making decisions. It communicates high-level commands (e.g., "product accepted") via a serial link to a real-time microcontroller like an ESP32. The microcontroller handles the low-level, time-sensitive tasks: reading the product arrival sensor, controlling the conveyor motor, actuating the rejector servo, and updating status LEDs. This distributed architecture separates high-level intelligence from real-time control, a common and robust pattern in industrial automation.

4 Results & Discussion

4.1 Experimental Setup

An experimental setup was developed and validated within a Docker environment. Ubuntu 22.04 and ROS 2 Humble was employed to create and execute the tests on the system. The AI validation pipeline was validated on a master dataset of over 50 products defined in a .csv file, along with corresponding label images designed to test several success and failure modes.

4.2 ML Model Performance

The core of our print quality check is a machine learning model tasked with classifying images of labels as "Good" or "Bad". We utilized the MobileNetV2 architecture [**?**], pre-trained on ImageNet, and fine-tuned it on our custom dataset of label images. The base convolutional layers were frozen to leverage their powerful feature extraction capabilities, while a new classification head was trained.

During initial testing, the model achieved a standard accuracy of **90.00%** using a default classification threshold of 0.5. However, for deployment in a manufacturing context, minimizing false negatives (i.e., incorrectly accepting a bad label) is paramount. We therefore analyzed the precision-recall curve and selected a more conservative operating threshold of **0.35**. This choice intentionally biases the model to be more sensitive to potential defects,

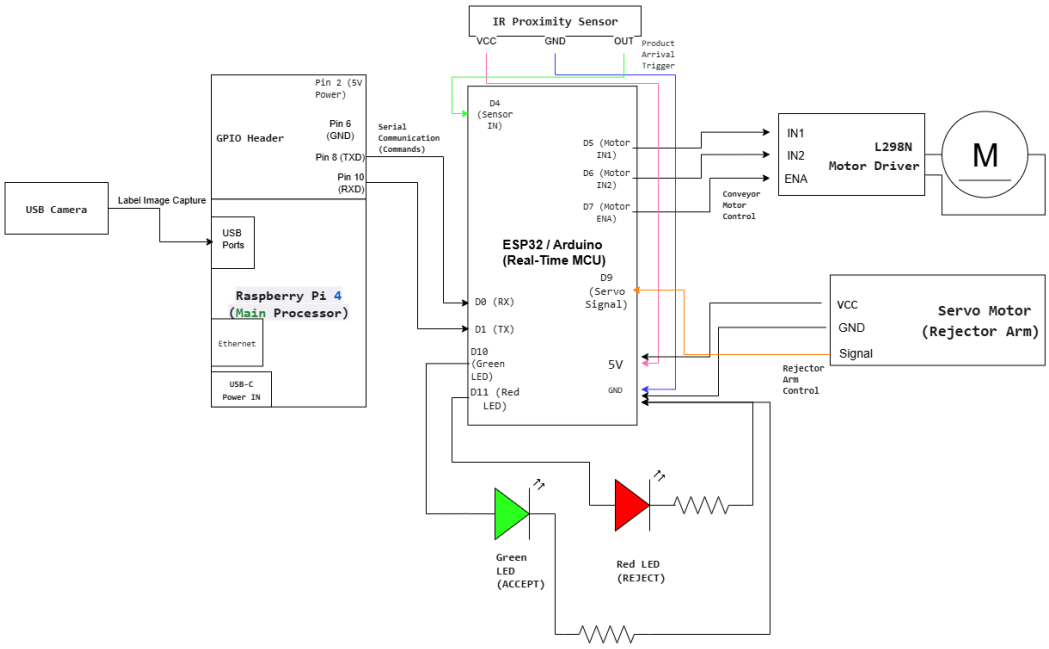

Figure 2: Conceptual hardware diagram showing the Raspberry Pi for high-level processing and an MCU for real-time control.

thereby increasing the recall for the "Bad Labels" class at the cost of overall accuracy. The detailed performance metrics at this deployed threshold are presented in Table 1.

Table 1: Classification Report for the MobileNetV2 Model at the Deployed Operating Threshold (0.35).

Class	**Precision**	**Recall**	**F1-Score**	**Support**
Bad Labels	1.00	0.47	0.64	15
Good Labels	0.65	1.00	0.79	15
Accuracy		**73.33%**		

The perfect precision score of 1.00 for the "Bad Labels" class is the key outcome of this tuning: at this threshold, the model *never* misclassifies a good label as a bad one. While the recall of 0.47 indicates it still misses some bad labels, this strategy ensures that any product flagged by the ML model is almost certainly defective, making it a reliable first-pass filter. The training history is illustrated in Fig. 3.

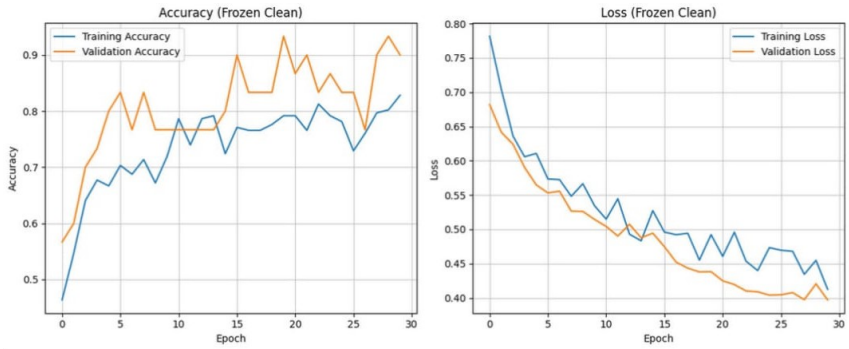

Figure 3: Model training history showing accuracy and loss.

4.3 End-to-End System Results

The ultimate test of the project is the performance of the integrated system, where the ML model, OCR validation, and ROS2 messaging work in concert. We conducted a series of end-to-end test runs simulating the full production cycle for a batch of electronic components. The system successfully sorted all test products into "Pass" or "Fail" bins based on the multi-stage validation logic.

Table 2 details the validation stages and provides specific examples of products that were correctly identified and rejected at each failure point. This demonstrates the system's ability to pinpoint the exact reason for a production error, which is invaluable for process improvement.

Table 2: System Validation Stages and Failure Mode Examples.

Validation Check	Description of Failure	Example DeviceID
QR Code Read	The QR code is unreadable due to printing errors, smudging, or damage.	**ELEC009**
ML Print Quality	The QR code is readable, but the ML model classifies the overall label image as "Bad" quality (e.g., skewed, faded).	**ELEC004**
OCR Text Validation	The text extracted from the label via OCR does not match the master data record for that DeviceID (e.g., wrong batch number).	**ELEC011**

The failure of device **ELEC004** was caught by the ML quality check, which flagged its label as poorly printed before the system proceeded to the data mismatch check. For device **ELEC011**, both the QR code and the print quality were acceptable, but the system correctly failed it because the batch ID printed on the label did not match the expected value in the master production CSV file. A sample view of the combined system at work is depicted in Fig. 4.

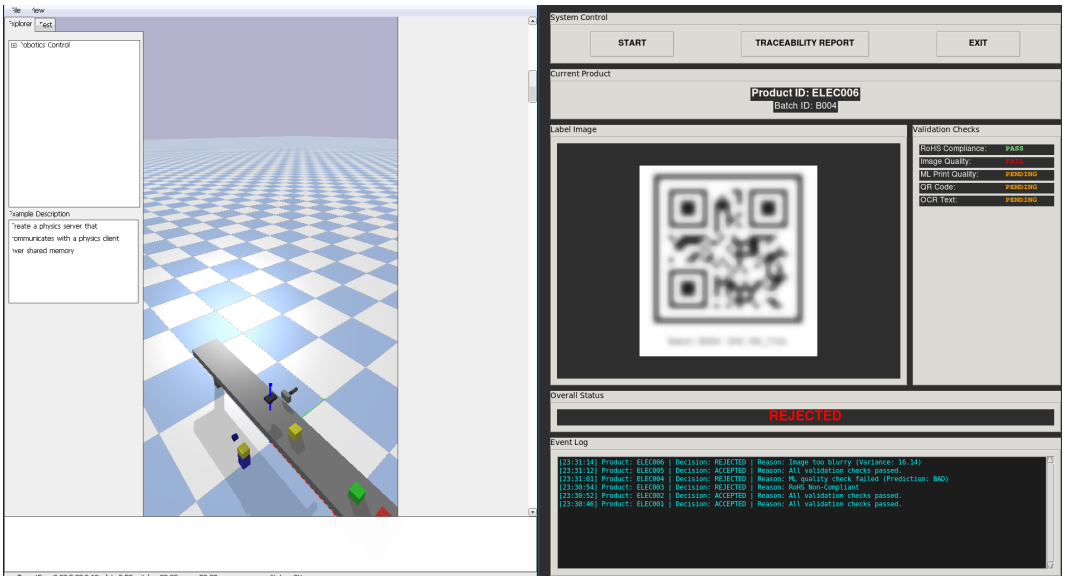

Figure 4: The integrated system showing the PyBullet simulation (left) and the Tkinter UI (right) during processing.

4.4 Discussion

The Experimental findings confirm that our combined system functions as designed. It successfully processes products in order and utilizes a multi-step validation logic, properly disposing of products from multiple data and image quality metrics. Successful early rejection of products (e.g., compliance, image quality) mirrors the efficiency of the pipeline. The high accuracy of the ML model trained for print quality mirrors the potential of employing custom deep learning solutions to individual-specific industrial inspection activities. The final traceability log has a full and exact audit trail for every processed unit.

5 Conclusion

In this project, we have effectively designed, implemented, and tested a complete Smart Product Labeling and Traceability System in the ROS 2 environment. Our suggested solution demonstrates a modular architecture with a PyBullet digital twin, different pre-trained AI models for QR and OCR verification, and a custom-trained ML model for label print quality inspection. The system supports real-time monitoring and control with a custom UI and ensures end-to-end traceability with a persistent SQLite database. The key limitations of this work are its reliance on a simulated environment and a controlled input image set. Future work can include deploying the system on real edge hardware (e.g., Raspberry Pi), enhancing AI robustness with more diverse training data and ROI-based OCR, and extending the web-based dashboard for analyzing the traceability data.

Acknowledgment

We would like to offer our sincere appreciation to Intel Corporation and the Intel Unnati scheme for this valuable industrial training program. We would like to thank our mentor, Dr. Starlet Ben Alex, for her continuous support and motivation during the project. We are also grateful to our college, Saintgits College of Engineering, for the foundational knowledge and resources that made this work possible.

Source Code and Assets

The complete source code for this project, including all ROS 2 packages, the Dockerfile for environment setup, training scripts for the machine learning model, and detailed usage instructions, is publicly available in our GitHub repository. All large assets, including the image datasets for training and evaluation, as well as the pre-trained machine learning model, are available for download via links provided in the repository's main README file.

GitHub Repository:
https://github.com/ShahazadAbdulla/ros2-intel-unnati-smart-labeling

References

[1] M. Raisul Islam, M. Zakir Hossain Zamil, M. Eshmam Rayed, M. Mohsin Kabir, M. F. Mridha, S. Nishimura, and J. Shin, "Deep learning and computer vision techniques for enhanced quality control in manufacturing processes," *IEEE Access*, vol. 12, pp. 121449–121479, 2024.

[2] J. Sanchez-Cubillo, J. Del Ser, and J. L. Martin, "Toward fully automated inspection of critical assets supported by autonomous mobile robots, vision sensors, and artificial intelligence," *Sensors*, vol. 24, no. 12, 2024.

[3] A. Saberironaghi, J. Ren, and M. El-Gindy, "Defect detection methods for industrial products using deep learning techniques: A review," *Algorithms*, vol. 16, no. 2, p. 95, 2023.

[4] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," 2017.

[5] E. Erős, M. Dahl, K. Bengtsson, A. Hanna, and P. Falkman, "A ros2 based communication architecture for control in collaborative and intelligent automation systems," *Procedia Manufacturing*, vol. 38, pp. 349–357, 2019.

[6] S. Ammendrup and L. Barcos, "The implementation of traceability systems," *Revue scientifique et technique*, vol. 25, no. 2, pp. 763–73, 2006.