INTEL UNNATI INDUSTRIAL TRAINING 2025
GITAM DEEMED TO BE UNIVERSITY

Problem Statement-5

Project Title

Smart Product Traceability Station for Labelling and Quality Verification

Team Name: IntelliTrace

Institution/Organization: Gandhi Institute of Technology and Management (GITAM)

Mentor Name: Dr. Sapna Jha

Team Members: Shreya Karmakar

Tejasri Anantapalli

Harini Mode

Date of Submission: July 10, 2025

1. Abstract

In this project we have the intention of achieving to add intelligence to modern manufacturing, this project simulates a smart, end-to-end product labeling and traceability system. Its mainly designed for small electronic products, the system identifies each unit, verifies essential parameters such as Device ID, Batch ID, Manufacturing Date, RoHS Compliance, and Serial Numbers through QR codes or barcodes which are on the product. It uses the top technologies like computer vision for capturing, OCR(Optical Character Recognition) to read text, and machine learning latest techniques to extract data from the labels, detect mismatches, and identify product defects in real-time modern manufacturing industries.

The system focuses on the logic set and automatically classifies products as accepted or rejected based on the quality checks, ensuring that only compliant and defect-free products units proceed. This project has the rejected products data logged and traced for further inspection. Each step, from product arrival to final decision-making, the data is recorded in a traceability log for better inspection, enabling complete data tracking of the product and auditability throughout the production process to not face any issues later.

By combining AI-driven validation which involves latest technologies, defect detection of the data of the dataset, and automated traceability, this project provides a software-centric solution that improves accuracy, quality control, and upgrades manufacturing workflows. It provides an intelligent approach to streamline labeling operations and maintain compliance in modern production environments.

2. Introduction

In today's modern manufacturing industries involving mainly electronics,medical,automotive demand high quality. To reach the standards, every product must have a label that solely identifies the product and stays till start to end of the production for better tracking. Accurate labeling allows for effective inventory management, defect tracking, product authentication, and regulatory compliance(like Rohs , FDA, etc). However, many current labeling solutions in factories are either manual or semi-automated which are not resulting in efficient results, including a high probability of human error, poor integration with quality assurance systems, and inadequate real-time data logging.

This project presents the design and simulation of a Smart Product Traceability Station—a unified, automated system that combines labeling, quality inspection, and data validation. In the project, the system uses computer vision to see the label, OCR (Optical Character Recognition) for reading the text, barcode/QR decoding to extract encoded information, and AI-driven verification logic to simulate a real-world industrial setup ensuring every unit is inspected, verified, and traceable throughout its lifecycle. On top, a machine learning model

trained on the MVTec ‘bottle’ dataset is used to detect surface or structural defects on the product based on the image of the product. Blending all these tools together in this project to simulate a real-world smart factory environment where every product is automatically inspected, verified, and made traceable throughout its manufacturing lifecycle providing better results.

2.1 Background

Modern manufacturing sectors—such as electronics, automotive, and medical devices—are under increasing pressure to deliver products that are not only high in quality but also fully traceable from production to delivery. Regulatory frameworks such as ISO standards, RoHS (Restriction of Hazardous Substances), and FDA guidelines necessitate precise labeling and verification mechanisms.

The standard way of labeling systems in factories operate independently of quality control mechanisms, often relying on human inspection or simple automation which end up with faults. These methods show a risk of mislabeling, missing data, and lack of integration with digital tracking systems to the product. Faults like these in an industry can lead to expensive product recalls, compliance violations, logistical inefficiencies and many other problems leading to less efficient processes.

To overcome these challenges, this project simulates a smart labeling and inspection station that uses machine vision to read labels using technologies like OCR, decode barcodes and QR codes of the product label, and verify that the extracted information matches expected metadata. The system ensures real-time logging of inspection results, batch IDs, serial numbers, and compliance status. By involving AI and ML technologies, the proposed solution flow improves the complete accuracy, consistency, and traceability in the production pipeline.

2.2 Objectives

The project's goal is to create a simulated smart station that can be used in the modern industries for better performance. We used different parameters of the product to ensure regulatory and quality compliance while carrying out automated product labeling and traceability tasks of the product. The system uses the latest AI and computer vision technologies to combine data logging, defect detection, and label verification. Among these the main goals are:

- Simulating a Product Labeling and Traceability System
  In this project, we developed a simulated environment that can be used in modern industries which specifically replicates an industrial product labeling process, involving main steps like computer vision, sensor data, and intelligent control to mimic real-world factory conditions.

- Verifying Key Product Metadata
  Our system does tasks like extracting data and verify essential information data printed or encoded on product labels, including:
    - Device ID – A unique product identifier of each product
    - Batch ID – Identifier that helps linking the product to its production batch,helps in traceability
    - RoHS Compliance Status – Confirmation of environmental regulation compliance rules of the product.
    - Manufacturing Date – Checking and validation of the date format and logic
- Applying and Validating QR/Barcode Labels
  We simulate label application with QR codes or barcodes and then decode them using selected software. The system compares the decoded data of the QR codes with expected set values (like Device ID or Batch ID) to check if the labeling is correct or incorrect. This step specifically helps in the step of ensuring traceability and preventing wrong or mismatched codes.
- Log Inspection Data for Traceability
  This step involves that every time a product is inspected, it will be recorded. The step of recording detailed inspection results involving timestamps, extracted data fields, defect status, and verification outcomes into structured logs for further auditing and traceability.
- Detect Label Mismatches and Visual Defects
  Used computer vision technologies and AI techniques to detect most occurred issues such as incorrect or mismatched label data, unreadable codes, misprints, and physical label defects,etc(e.g., misalignment or surface damage).This part helps to catch quality issues of the product.

3. Problem Definition and Life Cycle

Labelling, traceability, and compliance must be closely monitored in today's manufacturing. However, many of the existing solutions still have problems with workflows that don't work well together. Data logging, inspection, and labelling frequently take place independently. Errors, delays, and even problems with compliance may result from this. A smart, networked system that integrates automation, real-time checks, and more intelligent data management is required.

3.1 Problem Statement

In the manufacturing world these days, a lot of companies rely on semi-automated labeling systems. While these systems help in some ways, they still often fall victim to human mistakes and don't really integrate well with quality checks. This can lead to all sorts of issues, like labels being put on the wrong products, missing data, or important compliance details—like whether a product meets RoHS standards or its serial number—slipping through the cracks.

Our project aims to address these issues by simulating an intelligent, automated station that checks labels, verifies their accuracy, and logs all relevant data in real time. For this system, we intend to use computer vision technology driven by AI. It will verify crucial elements from the barcodes or QR codes, such as Device ID, Batch ID, Date of Manufacture, RoHS Compliance, and Serial Number, to ensure they are all accurate. If any units do not meet the standards, they will be flagged or discarded without delay. This way, we aim to boost the reliability and traceability of our products during manufacturing.

By automating these checks, companies can save themselves from the potential errors that might arise with systems operated by humans. Imagine a scenario in which every label is verified in an instant, ensuring accuracy and compliance, with not a lot of human noise in the background. This is happening right now in some assembly lines, where robots have taken over the jobs of the humans they replaced, and are doing those jobs better. All of this adds up to a smoother production cycle and keeps both manufacturers and consumers happy in the long run. This project represents a step toward improving the whole labeling and inspection process for industries that depend on such systems.

3.2 System Life Cycle

System Life Cycle A smart product labeling and inspection system functions in systems life cycle. Each unit goes through distinct steps of accurate labeling, verification, inspection for defects, and logging. These steps include:

1. Arrival of Product

The arrival of the product is either at the check station manually or on an automated conveyor belt.

2. Label Scanning and Metadata Extraction

Capture the label photo Employing OCR (Optical Character Recognition) and YOLOv5-based systems alongside barcode and QR code scanning to obtain critical details: Device ID Batch ID Date of Manufacture RoHS compliance status Serial Number

3.Data Validation

Checking the compliance and accuracy of data extraction involves confirming with reference standards in a backend database, or with certain rules like presence of batch ID and applicable RoHS compliances.

4. Label Application or Validation

In case there is no label for a product, one is created and “applied” virtually (or printed out physically). If there exists a label, it goes through checking processes to confirm it is accurate, complete, and understandable.

5. Defect Detection (CNN-Based)

Using a neural network that has been trained ahead of time, the system carries out visual defect detection. This process examines the physical aspects of the item such as scratches, missing components, and surface imperfections and visually classifies each item into valid or invalid categories.

6. Fault Handling and Rejection.

If at any point of time a defect, formatting issue or mismatch is identified (via metadata or visual inspection) the system flags the product as rejected and sends it to the rejected products log.

7. Traceability Logging.

All of our traceability systems are used for the logging of the inspection data which includes:

- Extracted metadata
- OCR results
- CNN classification results
- Pass/fail decision
- Timestamp
- Reason for rejection (if any)

This log is for full traceability in quality control and audits.

4. System Overview:

This project implements an intelligent inspection and labeling station, developed using software tools, depicting specific significant characteristics of classification systems used currently in modern manufacturing. Of note, the operation of the developed system herein involves the following:

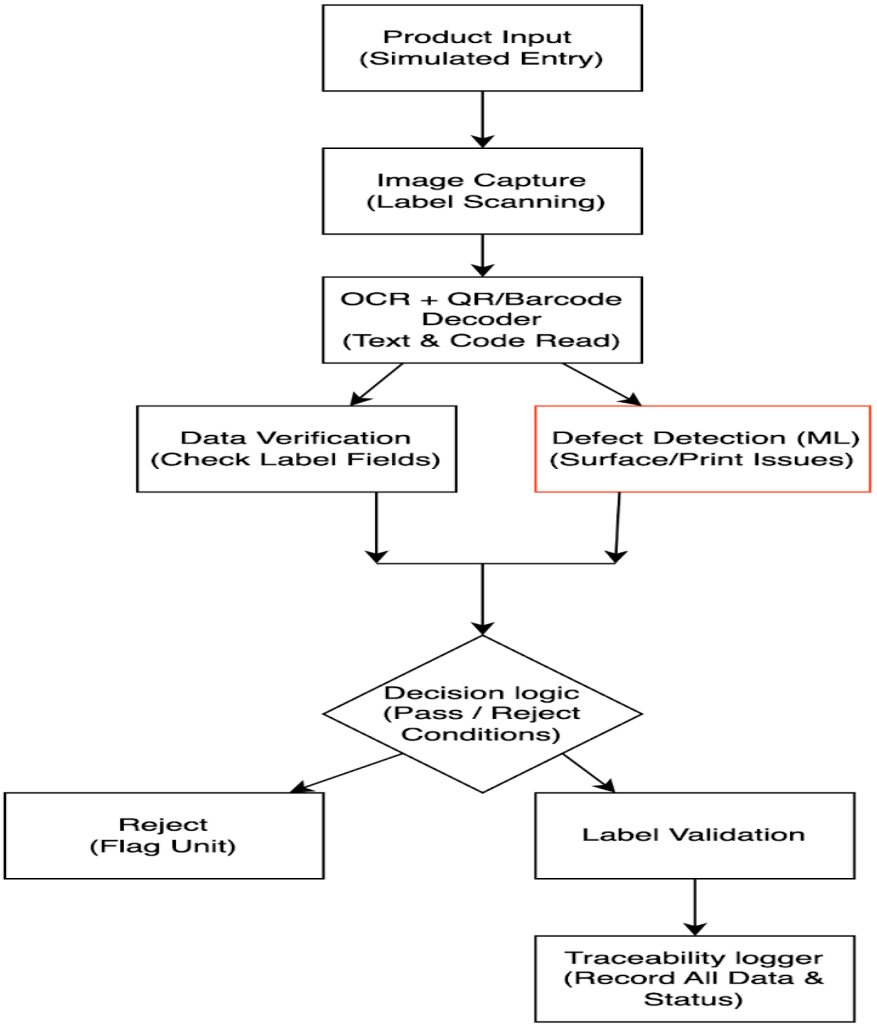
- **Simulated Conveyor Input:** Products enter through a spooled conveyor belt mimicking an uninterrupted production line.

- **Label Image Capture:** Image input captures each product's label, which will later be used in inspecting and extracting data.
- **Metadata Extraction using OCR:** The image is then subjected to Optical Character Recognition techniques to extract metadata, including Device ID, Batch ID, Manufacturing Date, and RoHS compliance information.
- **Compliance Verification:** The obtained metadata is verified against set parameters and reference data to determine whether the product complies with prescribed regulations and quality assurance.
- **QR/Barcode Generation and Validation:** The application can either generate QR codes or barcodes for labeling or decode already encoded information to verify serial numbers and other data encoded.
- **Computer Vision-Based Defect Detection:** Computer vision's advanced algorithms will detect any visible defects in the misprinted labels, damaged, or illegible codes.
- **Rejection of Non-Compliant Units:** Any products that do not comply with any inspection or verification step are identified and simulated to be rejected in the production flow.
- **Traceability Logging:** Each inspection result, verification status, and action taken by the system is recorded orderly to enable traceability and auditing.

This software development illustrates how integrated AI and Automation contribute to accurate labeling, quality control, and traceability in manufacturing processes.

5. System Architecture

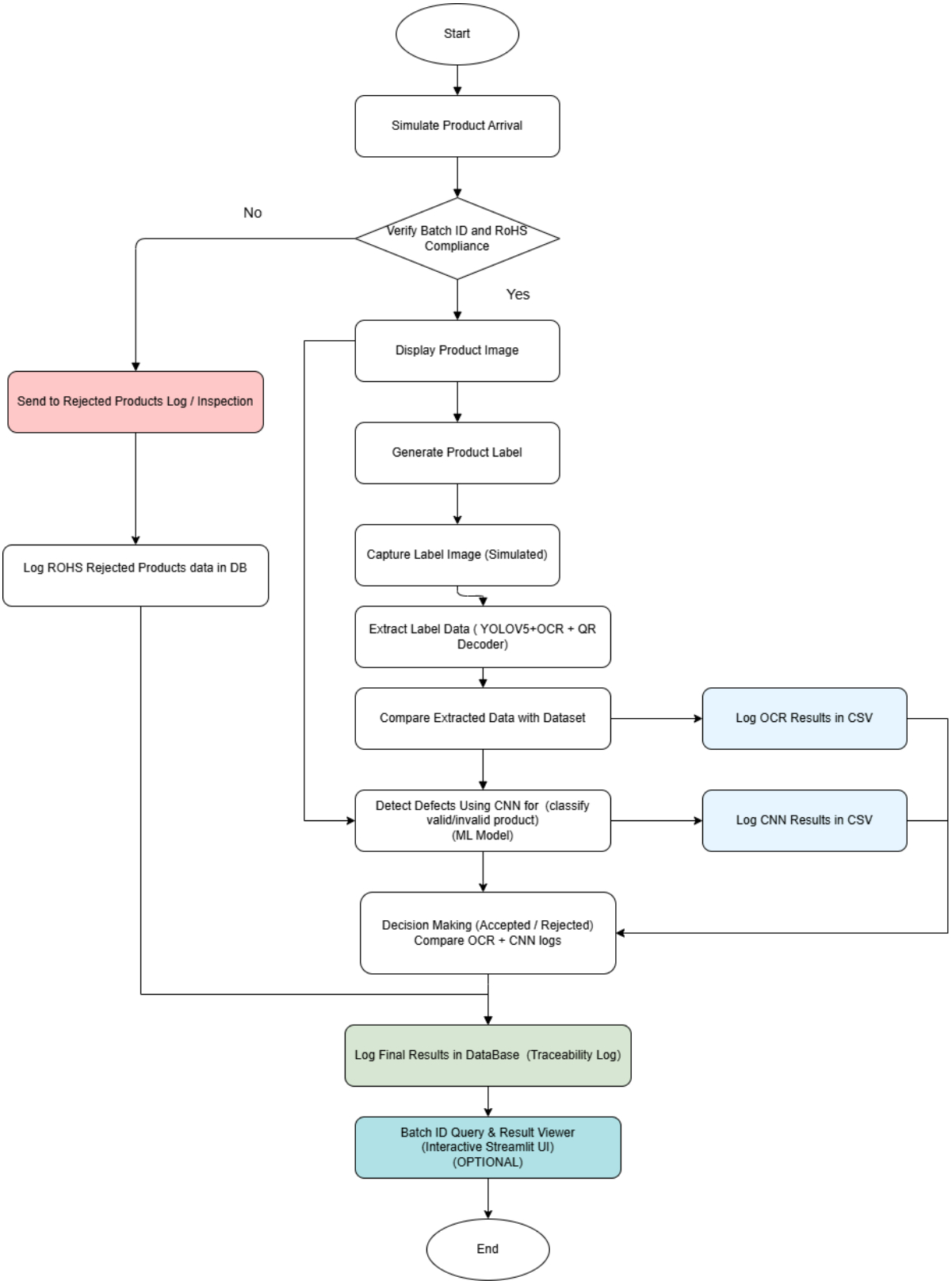
5.1 Block Diagram



Smart Labeling & Traceability System Architecture

The above project's block diagram explains the flow of the project. It specifically focuses on how the product inspection and labeling system operates in the entire process. The initial step is a product entry is done first, and then an image is taken of the product and the label is read using OCR technology and QR decoding. After the data is extracted, the next step is the extracted data is checked against the predicted values. In parallel, a machine learning model detects defects to find surface or label problems. A decision logic we created to solve the problem determines whether the unit passes or fails based on the two verified results, initiating the proper logging, labeling, or rejection procedures. Complete traceability and adherence to manufacturing standards are thus guaranteed.

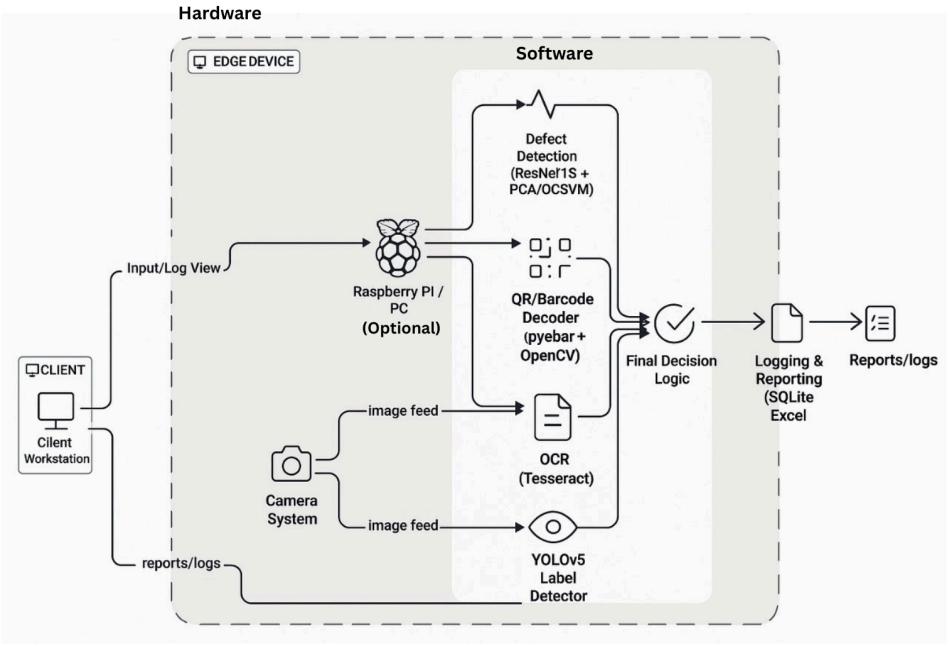
5.2 Flow Chart



The flowchart outlines a smart product traceability and inspection system. It starts with simulating product arrival, followed by verifying the Batch ID and RoHS compliance. If the product fails this check, it is logged as rejected and then added to the rejected products

database. If the product passes, the system displays the product image, generates a label, and simulates label image capture. The label data is then extracted using YOLOv5, OCR, and a QR decoder, and compared with the reference dataset. In parallel, defects are detected using a CNN model to classify products as valid or invalid. OCR and CNN results are logged separately in CSV files. Based on a comparison of both results, the system makes a final decision (approved or rejected) and logs the result in the traceability database. The process ends after final logging.

5.3 Hardware / Software Mapping



Deployment Diagram

Hardware–Software Mapping Based on Diagram

Hardware Component	Mapped Software Function	Explanation
Client Workstation	- View Input Logs - Access Reports (from SQLite/Excel)	Used by operators to monitor or retrieve reports and logs. Connects to the edge device.
Camera System	- Image Feed Input to: • YOLOv5 Label Detector • OCR (Tesseract) • QR/Barcode Decoder	Captures images of the product and its label for processing by all modules.

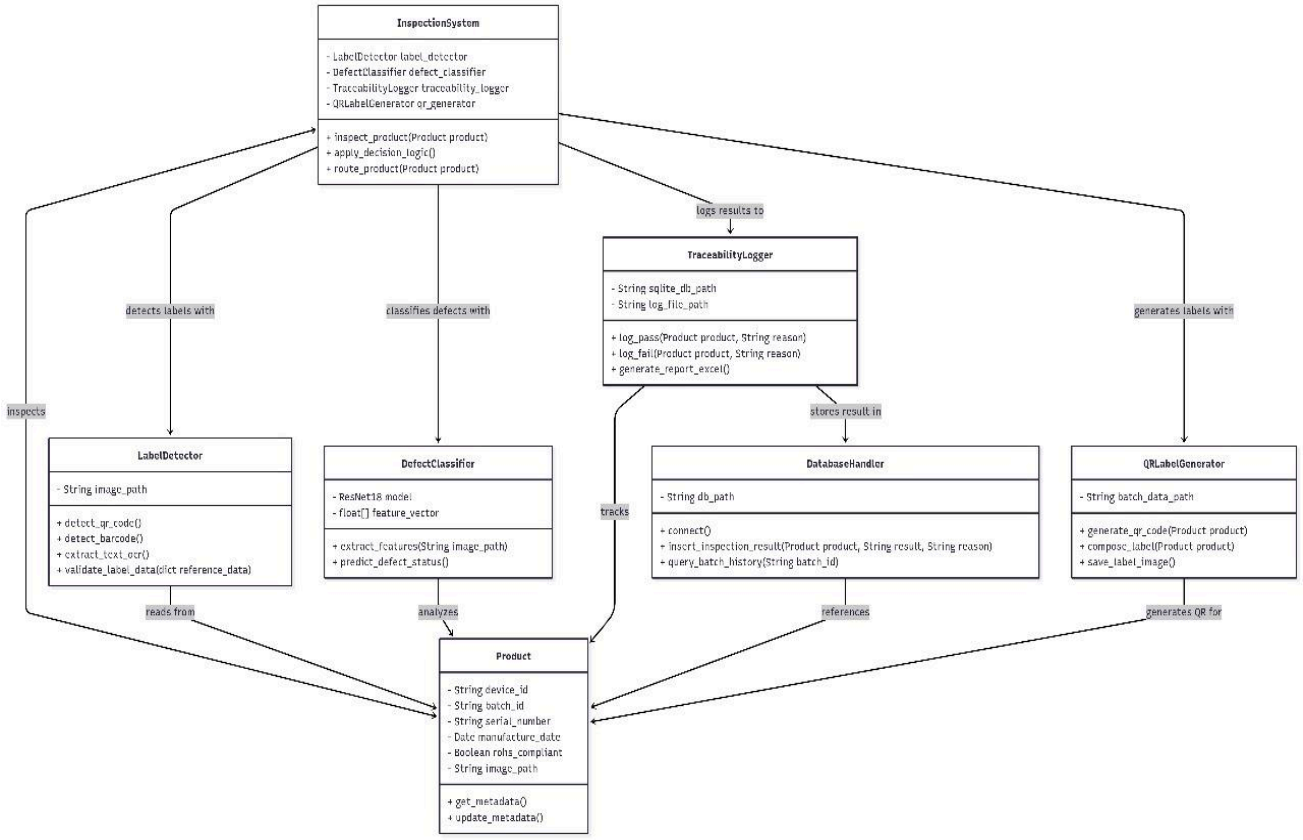
Raspberry Pi / PC (Optional)	- Hosts all the software modules It includes: • Defect Detection (ResNet + PCA/OCSVM) • OCR • QR/Barcode Decoder • YOLOv5 Label Detection • Final Decision Logic • Logging & Reporting (SQLite/Excel)	This basically acts as the edge computing unit that performs all AI processing and reporting logic.
Storage (Pi/PC)	- Logs and stores: • Image files • Decision outputs • SQLite DB • Excel /CSV reports	Holds all system-generated outputs.
Display Monitor (optional)	- GUI display for real-time inspection results (if its integrated)	For viewing live feedback on-site.

Software Modules as per Diagram

Software Module	Runs On	Function
Defect Detection (ResNet + PCA/OCSVM)	Raspberry Pi / PC	Product images will be analyzed for anomalies or defects.
QR/Barcode Decoder (pyzbar + OpenCV)	Raspberry Pi / PC	Batch ID, Serial number is drawn out from the QR/barcodes.
OCR (Tesseract)	Raspberry Pi / PC	OCR reads printed text such as dates, RoHS and serial code.

YOLOv5 Label Detector	Raspberry Pi / PC	Captures and processes images containing labels or stickers on product packaging to identify them.
Final Decision Logic	Raspberry Pi / PC	Collaboration logic based on the outputs of the last three modules is processed here.
Logging & Reporting (SQLite, Excel)	Raspberry Pi / PC	Document all inspection decisions and produce report results in excel or sqlite database.
Input/Log View (Client)	Client Workstation	The client can issue commands through the edge device or view logs through this interface.

5.4 Class Diagram



InspectionSystem handles label reading, defect detection, QR code creation, and result logging.

It uses **LabelDetector** to scan metadata, **DefectClassifier** for examining image flaws, and **QRCodeLabelGenerator** for label creation.

Results go to **TraceabilityLogger** and then to **DatabaseHandler**.

Product stores all required details like device ID, batch ID,serial_number,image_path,rohs_compliant

6. Software / Hardware Requirements

6.1 Software

Software/Tool	Description
Python 3.8+	Primary programming language for system logic, model inference, and processing
YOLOv5	Object detection model for identifying the label components
EasyOCR	Optical Character Recognition library to read text from label images
Pyzbar	QR code decoder used to extract encoded serial data from labels
OpenCV	Image processing library for image reading, pre-processing
Pandas / NumPy	Data manipulation, validation, and comparison with reference dataset
Matplotlib (optional)	Visual representation of logs or graphs (if required)
SQLite3	Local lightweight database for inspection result logging
Streamlit / Tkinter (optional)	GUI framework for user interface and interaction

VS Code / Jupyter / Google Colab / PyCharm	IDE or notebook environment for development and testing
OS	Windows 10 or later (Tested); supports cross-platform

6.2 Hardware

Hardware Component	**Specification / Description**
PC / Laptop	Handles all AI and processing, decision-making, including data logging tasks in your project. (Here it replaces physical hardware like Arduino, sensors, or printers by simulating those steps with software modules.)
Webcam or USB Camera	Optional: For capturing product/label images (in simulation, static images used)
Storage	Minimum 2GB free disk space for images, logs, models
Display Monitor(Optional)	For visualizing GUI output and inspection logs
Mouse & Keyboard	Standard input devices for interacting with GUI or IDE

Note: No external microcontroller (e.g., Arduino/ESP32) or label printer is used in this implementation; label generation and product verification are fully simulated using image files in our project.

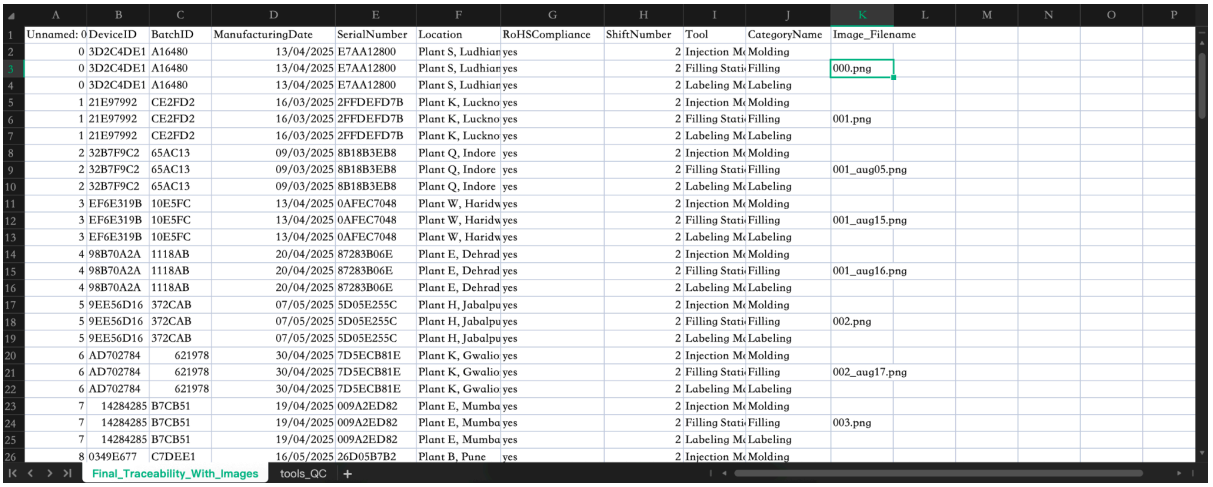
7. Implementation Phase

Dataset Creation:

This dataset is created for smart product labeling and traceability systems. Involved data like Device ID of the product , Batch ID of the product , and Serial Number of the product, checks compliance (RoHS), location of the product production, shift number of the product production, tools used for the product(focused on main three tools),and the filename of the

image linked to the product. Each entry is linked with the image file name for the inspection process ensuring complete traceability across the production pipeline.

1. Dataset of the final_traceability of the products :

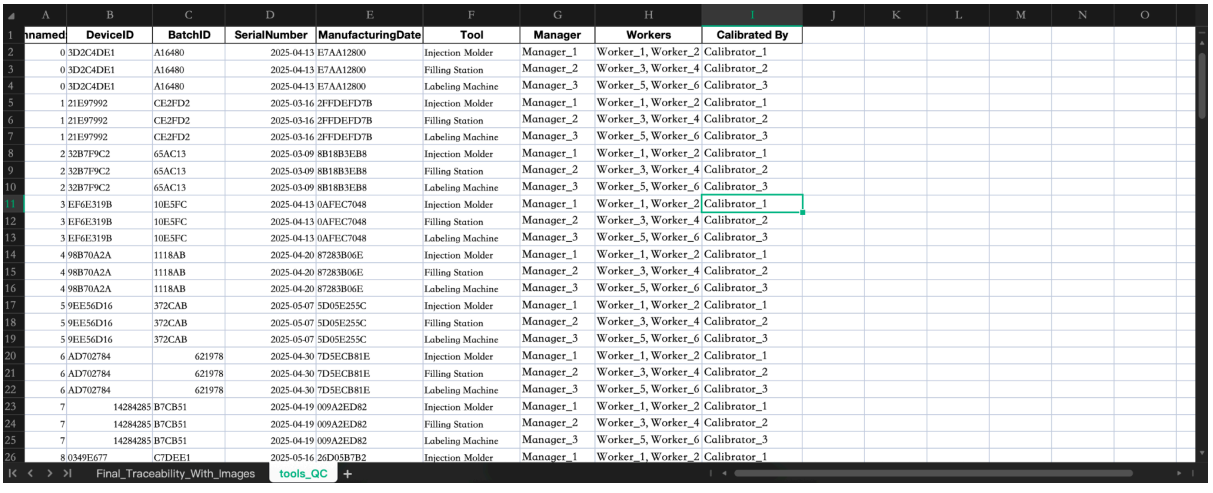


	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	Unnamed: 0	DeviceID	BatchID	ManufacturingDate	SerialNumber	Location	RoHSCompliance	ShiftNumber	Tool	CategoryName	Image_Filename					
2	0	3D2C4DE1	A16480	13/04/2025	E7AA12800	Plant S, Ludhian	yes	2	Injection Mc	Molding						
3	0	3D2C4DE1	A16480	13/04/2025	E7AA12800	Plant S, Ludhian	yes	2	Filling Stati	Filling	000.png					
4	0	3D2C4DE1	A16480	13/04/2025	E7AA12800	Plant S, Ludhian	yes	2	Labeling Mc	Labeling						
5	1	21E97992	CE2FD2	16/03/2025	2FFDEFD7B	Plant K, Luckno	yes	2	Injection Mc	Molding						
6	1	21E97992	CE2FD2	16/03/2025	2FFDEFD7B	Plant K, Luckno	yes	2	Filling Stati	Filling	001.png					
7	1	21E97992	CE2FD2	16/03/2025	2FFDEFD7B	Plant K, Luckno	yes	2	Labeling Mc	Labeling						
8	2	32B7F9C2	65AC13	09/03/2025	8B18B3EB8	Plant Q, Indore	yes	2	Injection Mc	Molding						
9	2	32B7F9C2	65AC13	09/03/2025	8B18B3EB8	Plant Q, Indore	yes	2	Filling Stati	Filling	001_aug05.png					
10	2	32B7F9C2	65AC13	09/03/2025	8B18B3EB8	Plant Q, Indore	yes	2	Labeling Mc	Labeling						
11	3	EF6E319B	10E5FC	13/04/2025	0AFEC7048	Plant W, Haridw	yes	2	Injection Mc	Molding						
12	3	EF6E319B	10E5FC	13/04/2025	0AFEC7048	Plant W, Haridw	yes	2	Filling Stati	Filling	001_aug15.png					
13	3	EF6E319B	10E5FC	13/04/2025	0AFEC7048	Plant W, Haridw	yes	2	Labeling Mc	Labeling						
14	4	98B70A2A	1118AB	20/04/2025	87283B06E	Plant E, Dehrad	yes	2	Injection Mc	Molding						
15	4	98B70A2A	1118AB	20/04/2025	87283B06E	Plant E, Dehrad	yes	2	Filling Stati	Filling	001_aug16.png					
16	4	98B70A2A	1118AB	20/04/2025	87283B06E	Plant E, Dehrad	yes	2	Labeling Mc	Labeling						
17	5	9EE56D16	372CAB	07/05/2025	5D05E255C	Plant H, Jabalpu	yes	2	Injection Mc	Molding						
18	5	9EE56D16	372CAB	07/05/2025	5D05E255C	Plant H, Jabalpu	yes	2	Filling Stati	Filling	002.png					
19	5	9EE56D16	372CAB	07/05/2025	5D05E255C	Plant H, Jabalpu	yes	2	Labeling Mc	Labeling						
20	6	AD702784	621978	30/04/2025	7D5ECB81E	Plant K, Gwalio	yes	2	Injection Mc	Molding						
21	6	AD702784	621978	30/04/2025	7D5ECB81E	Plant K, Gwalio	yes	2	Filling Stati	Filling	002_aug17.png					
22	6	AD702784	621978	30/04/2025	7D5ECB81E	Plant K, Gwalio	yes	2	Labeling Mc	Labeling						
23	7	14284285	B7CB51	19/04/2025	009A2ED82	Plant E, Mumba	yes	2	Injection Mc	Molding						
24	7	14284285	B7CB51	19/04/2025	009A2ED82	Plant E, Mumba	yes	2	Filling Stati	Filling	003.png					
25	7	14284285	B7CB51	19/04/2025	009A2ED82	Plant E, Mumba	yes	2	Labeling Mc	Labeling						
26	8	0349E677	C7DEE1	16/05/2025	26D05B7B2	Plant B, Pune	yes	2	Injection Mc	Molding						

Final_Traceability_With_Images | tools_QC | +

2. Dataset of the tools_Quality_check of the products :

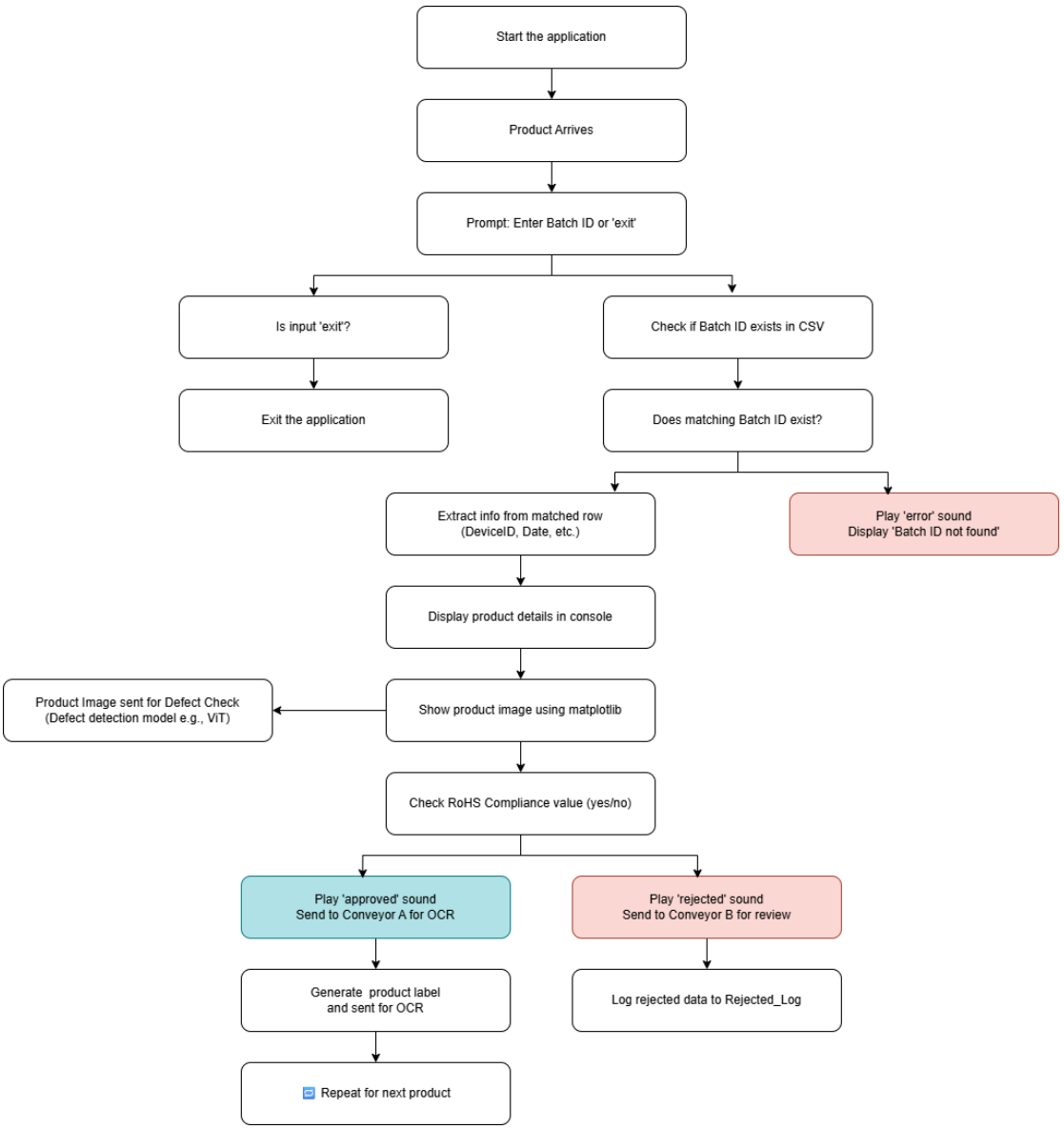
The next dataset we created is of the tools quality check, it involves main columns like Device ID of the product , Batch ID of the product , and Serial Number of the product, tools of the product , manager of the product,workers of the product production and the calibrator of the product. All these details are used for the traceability of the product. If any product has to be traced first reference will be for the final traceability data sheet and for the deep inspection of the product the tools quality check will be referred for efficient traceability results.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	nnamed	DeviceID	BatchID	SerialNumber	ManufacturingDate	Tool	Manager	Workers	Calibrated By						
2	0	3D2C4DE1	A16480	2025-04-13	E7AA12800	Injection Molder	Manager_1	Worker_1, Worker_2	Calibrator_1						
3	0	3D2C4DE1	A16480	2025-04-13	E7AA12800	Filling Station	Manager_2	Worker_3, Worker_4	Calibrator_2						
4	0	3D2C4DE1	A16480	2025-04-13	E7AA12800	Labeling Machine	Manager_3	Worker_5, Worker_6	Calibrator_3						
5	1	21E97992	CE2FD2	2025-03-16	2FFDEFD7B	Injection Molder	Manager_1	Worker_1, Worker_2	Calibrator_1						
6	1	21E97992	CE2FD2	2025-03-16	2FFDEFD7B	Filling Station	Manager_2	Worker_3, Worker_4	Calibrator_2						
7	1	21E97992	CE2FD2	2025-03-16	2FFDEFD7B	Labeling Machine	Manager_3	Worker_5, Worker_6	Calibrator_3						
8	2	32B7F9C2	65AC13	2025-03-09	8B18B3EB8	Injection Molder	Manager_1	Worker_1, Worker_2	Calibrator_1						
9	2	32B7F9C2	65AC13	2025-03-09	8B18B3EB8	Filling Station	Manager_2	Worker_3, Worker_4	Calibrator_2						
10	2	32B7F9C2	65AC13	2025-03-09	8B18B3EB8	Labeling Machine	Manager_3	Worker_5, Worker_6	Calibrator_3						
11	3	EF6E319B	10E5FC	2025-04-13	0AFEC7048	Injection Molder	Manager_1	Worker_1, Worker_2	Calibrator_1						
12	3	EF6E319B	10E5FC	2025-04-13	0AFEC7048	Filling Station	Manager_2	Worker_3, Worker_4	Calibrator_2						
13	3	EF6E319B	10E5FC	2025-04-13	0AFEC7048	Labeling Machine	Manager_3	Worker_5, Worker_6	Calibrator_3						
14	4	98B70A2A	1118AB	2025-04-20	87283B06E	Injection Molder	Manager_1	Worker_1, Worker_2	Calibrator_1						
15	4	98B70A2A	1118AB	2025-04-20	87283B06E	Filling Station	Manager_2	Worker_3, Worker_4	Calibrator_2						
16	4	98B70A2A	1118AB	2025-04-20	87283B06E	Labeling Machine	Manager_3	Worker_5, Worker_6	Calibrator_3						
17	5	9EE56D16	372CAB	2025-05-07	5D05E255C	Injection Molder	Manager_1	Worker_1, Worker_2	Calibrator_1						
18	5	9EE56D16	372CAB	2025-05-07	5D05E255C	Filling Station	Manager_2	Worker_3, Worker_4	Calibrator_2						
19	5	9EE56D16	372CAB	2025-05-07	5D05E255C	Labeling Machine	Manager_3	Worker_5, Worker_6	Calibrator_3						
20	6	AD702784	621978	2025-04-30	7D5ECB81E	Injection Molder	Manager_1	Worker_1, Worker_2	Calibrator_1						
21	6	AD702784	621978	2025-04-30	7D5ECB81E	Filling Station	Manager_2	Worker_3, Worker_4	Calibrator_2						
22	6	AD702784	621978	2025-04-30	7D5ECB81E	Labeling Machine	Manager_3	Worker_5, Worker_6	Calibrator_3						
23	7	14284285	B7CB51	2025-04-19	009A2ED82	Injection Molder	Manager_1	Worker_1, Worker_2	Calibrator_1						
24	7	14284285	B7CB51	2025-04-19	009A2ED82	Filling Station	Manager_2	Worker_3, Worker_4	Calibrator_2						
25	7	14284285	B7CB51	2025-04-19	009A2ED82	Labeling Machine	Manager_3	Worker_5, Worker_6	Calibrator_3						
26	8	0349E677	C7DEE1	2025-05-16	26D05B7B2	Injection Molder	Manager_1	Worker_1, Worker_2	Calibrator_1						

Final_Traceability_With_Images | tools_QC | +

7.1 Product Arrival and Verify compliance



Configuration and Setup includes

- Required imports: Pandas for organizing data
- OS for managing files
- PIL for processing images
- Winsound for audio feedback
- Numpy and Matplotlib.

Set up folders for:

- Product metadata CSV file
- Imaging folder
- Sounds files folder
- Log CSV folder
- Matplotlib images outputs folder

1. System Initialization and Input Phase where the system starts and awaits product input.

Each arriving product is acknowledged and the user is prompted to enter the associated Batch ID.If the user types "exit", the application closes gracefully.

2. Show Product Image Utility A helper function show_image() is defined to display product images via matplotlib which eases displaying images in forms of charts or plots.

3. Main Loop For Product Processing Continuous loop allows performing verification on multiple products until the user inputs “exit.”

- Batch ID Input & Search Through CSV Using A User Given Identifier User is to input Batch ID through designated box interface then System verifies existence of entered ID in the associated CSV file If not found system plays an error sound and shows relevant message
- Extract and Display Metadata

  If it is located, retrieves the following metadata:

  Device ID, Manufacturing Date, Serial Number and RoHS compliance.

  Show the console output of product information.

- Show Product Image

  Provided an image filename and it is valid:

  Shows the image from the specified directory as well as loads it.

  If not found, issue a warning.

4. Validation of RoHS Compliance and Workflow Control

- RoHS checks the product to see if it meets required specifications:

  Beep sound for approval.

  Display prints “Send to Conveyor A for OCR.”

- For non-compliance:

  Rejection sound for no response.

  Display prints “Send to Conveyor B for review.”

  Skip to rejection logging.

5. Log Rejections

- In cases where a product is rejected:

  Add a column with Validation Status labeled 'Rejected'.

Append this entry along with : Rejected log.csv

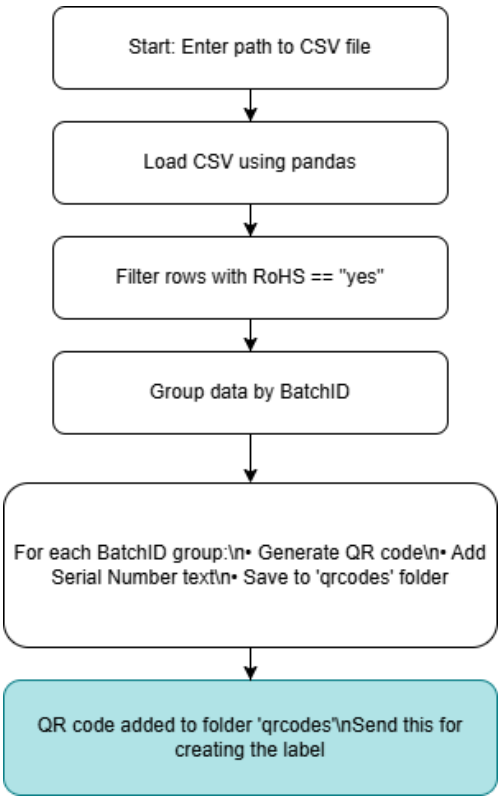
Duplicate checks are enabled so that multiple logs are not made into one stream

7.2 Creation of label

QR Code + Product Metadata + Label Layout Design = Final Label Creation

Initially, we produced the QR code using the qrcode library, which included the basic and volume relevant product details (e.g. Device ID, Batch ID, Serial Number, etc.). Once the QR code image was created and saved, we used the PIL (Python Imaging Library) to format the final label using the QR Code image. We'll use other required metadata, such as the product name, manufactured date, and compliance information (e.g., RoHS), and include that as the text on the label. The final label image will then be saved for print logging.

1. Generation of QR Codes:



- The QR code generation subsystem of the system in focus is written entirely in Python with the following libraries as key dependencies:
- pandas:For managing and obtaining information from the CSV files.
- qrcode:Used to generate QR codes based on batch data.
- PIL (Pillow): Used for inscribing Serial Numbers text below QR code images.

- Also, os and json – for tree structure operations and containing text formatted as structured data.

Overview of processes:

- Loading CSVs: The system picks product batches from .csv files in a specified path by using pandas.read_csv() method.
- RoHS Filtering: Only rows marked “yes” under RoHSCompliance are kept for QR creation.
- Group By Batch: Data is grouped according to BatchID so as to have one QR code per batch.

QR Code Generation:

- The following core fields are defined (DeviceID, BatchID, ManufacturingDate, SerialNumber, Location, RoHSCompliance).
- They will be extracted and then serialized into a compact JSON string format that preserves only essential structures.
- Using the qrcode library, this JSON shall then be encoded into a corresponding QR code.

Image annotation Pillow processes:

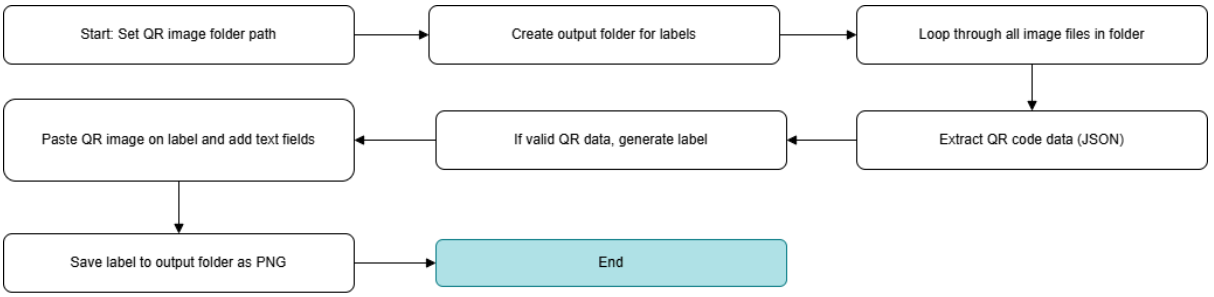
- QR Code generation yields an image with its respective caption which can be augmented further vertically.
- As stated before, serial numbers will also be inscribed beneath their respective codes using pillow functions.
- File saving Each final image is appended within a local directory tagged qrcodes under “qrcode_batch_<BatchID>.png” naming convention .

Expected Results :

- There exists a folder named qrcodes/ which contains all coded images each depicting a singular batch alongside its serial number caption.
- The images are now prepared for use in establishing labels and for the packaging of the product.

QR Code of the Product

2. Creation of Label:



Logic:

The label generation module takes product QR images as input and outputs final labeled images that include both machine-readable (QR) and human-readable information.

Tools & Libraries Used:

- OpenCV – For reading images to decode QR codes.
- pyzbar – To extract embedded JSON data from QR codes.
- Pillow (PIL) – To design the final label image with text and QR placement.
- OS/ Json – For folder handling and data parsing.

Step-by-Step Workflow:

The system scans a specified folder for all QR code images (PNG/JPG).

Each image is decoded using pyzbar to extract the embedded JSON data.

From this data, key fields are extracted such as:

- Device ID, Batch ID, Serial Number
- Manufacturing Date, Location
- RoHS Compliance status

A clean label is generated using Pillow:

The label includes static text (Product Name, Description).

Dynamic fields from the QR are rendered as readable text.

The QR image is resized and pasted onto the label.

Each label is saved in the final_labels/ folder as Label_<BatchID>.png.

Error Handling:

If QR decoding fails or JSON format is invalid, the script logs a clear message and skips that image.

Expected Results:

- High-quality label images with both QR code and product metadata.
- Ready to print and affix on electronic components for traceability.

CrystalClear Springs
Natural Spring Water - 500 mL
Bottled by CrystalClear Inc., Pittsburgh, PA
Source: Blue Ridge Springs, PA

MFG Date: 30-03-2025
EXP Date:
Batch ID: 21224

BPA Free | Recyclable | NSF Certified
Scan the QR code to verify authenticity.

7.3 Metadata Extraction(OCR)

Integrating the OCR pipeline resolves issues related to extracting and validating product metadata from labels. The bespoke pipeline combines object detection using YOLOv5, QR decoding via Pyzbar, and text extraction through Tesseract OCR.

Dataset Input

The algorithm is tested on orderly structured datasets with images of product labels.

Each image has:

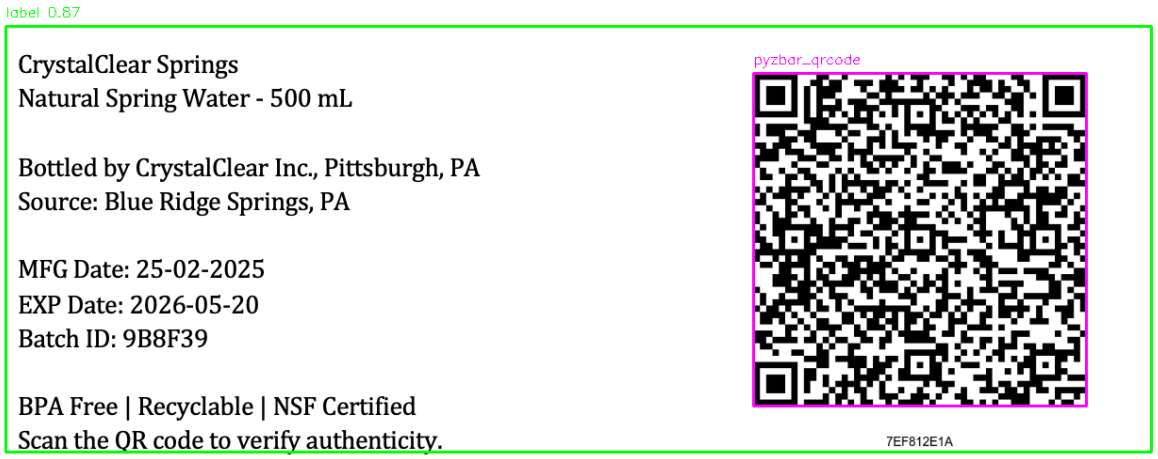
- A QR code embedded with JSON metadata
- A text label containing human-readable manufacturing information

i. Object Detection using YOLOv5

Our custom-trained YOLOv5 model was fine-tuned and loaded via the **torch.hub** interface and includes **model classes**:

1. Class 0: QR Code
2. Class 1: Product Label

The model locates QR codes and labels from images in the image dataset **'label_images'**, outputting bounding boxes over the input images and stores them in a folder named **'final_bounded_images'** as follows



ii. QR Code Decoding (Activated through pyzbar)

The QR region decoded using Pyzbar returned:

- Device ID
- Batch ID
- Manufacturing Date
- RoHS Compliance

Fallback decoding directly kicks in when the YOLOv5 custom model cannot detect the QR code due to blur or other environmental factors.

iii. OCR Implementation via Tesseract

The relevant portions of detected labels are sent to Tesseract OCR for recognition. **Further preprocessing steps include:**

- Conversion to grayscale
- Image denoising
- Adaptive thresholding

Post-extraction, the identified text was refined to eliminate extraneous characters for the key fields that needed to be filled in.

iv. Data Structuring and Logging

The following will be recorded for each detection in a structured CSV file.

- Timestamp
- Device ID
- Batch ID
- Image Path
- Source: QR / OCR
- Status: Pass/Reject Decision
- Extracted label text

The CSV would be generated based on:

- OCR confidence
- Valid metadata
- Blurriness score

Some Example Technologies and Libraries Used:

- Object detection with YOLOv5 (PyTorch)
- OCR with Tesseract OCR
- QR decoding with pyzbar + OpenCV
- image processing with OpenCV and PIL.
- Data logging and output were done using pandas and datetime

Outcome

- Reliably parses machine-readable (QR) and human-readable (label) data.
- Fallback in detecting and decoding QR codes using the Pyzbar library.
- Scalable logging of metadata for traceability and auditing.

Results logged into total_label(1).csv

Timestamp	Device ID	Batch ID	Status	Image	Source	Raw Data
21-06-2025 15:05	049088CD	15C215	Passed	/content/drive/MyDrive/label_in	QR/Barcode	DeviceID: 049088CD, BatchID: 15C215, ManufacturingDate: 10-03-2025, SerialNumber: E9C808667, Location: Plant C, Hyderabad, RoHSCompliance: yes
21-06-2025 15:05			Rejected	/content/drive/MyDrive/label_in	OCR	OG EEE IED IEEE EE EES rs -= - re m7 Natural Spring Water - 500 mL | art 7 rh [oa] Sr at eae Bottled by CrystalClear Inc., Pittsburgh, PA en he Shah oa ts ai . a ee a
21-06-2025 15:05	2FC18128	738105	Passed	/content/drive/MyDrive/label_in	QR/Barcode	DeviceID: 2FC18128, BatchID: 738105, ManufacturingDate: 13-05-2025, SerialNumber: C088A3F31, Location: Plant O, Visakhapatnam, RoHSCompliance: yes
21-06-2025 15:05		738105	Passed	/content/drive/MyDrive/label_in	OCR	CrystalClear Springs 1. Piss Natural Spring Water - 500 mL fe] Si Sr ety [a] WE reals red a Bottled by CrystalClear Inc, Pittsburgh, PA Iz ny HT Lid Source: Blue Ric
21-06-2025 15:05	3441F704	548077	Passed	/content/drive/MyDrive/label_in	QR/Barcode	DeviceID: 3441F704, BatchID: 548077, ManufacturingDate: 09-05-2025, SerialNumber: 1B13903DF, Location: Plant R, Patna, RoHSCompliance: yes
21-06-2025 15:05		548077	Passed	/content/drive/MyDrive/label_in	OCR	UrystalUlear oprings . <omms Natural Spring Water - 500 mL [oj aan Th bony [oj [Speech ttn aes Bottled by CrystalClear Inc., Pittsburgh, PA aL u my na * Source
21-06-2025 15:05	6DCBAB9F	476AD4	Passed	/content/drive/MyDrive/label_in	QR/Barcode	DeviceID: 6DCBAB9F, BatchID: 476AD4, ManufacturingDate: 28-02-2025, SerialNumber: 9BF9DE5C8, Location: Plant X, Chandigarh, RoHSCompliance: yes
21-06-2025 15:05		476AD4	Passed	/content/drive/MyDrive/label_in	OCR	Urystaulear springs . = Natural Spring Water - 500 mL [a] i apne 1 .[c] a Rant Salo Bottled by CrystalClear Inc., Pittsburgh, PA Tie ph. - I Source: Blue Ridge Spring
21-06-2025 15:05	8F613DE3	37271A	Passed	/content/drive/MyDrive/label_in	QR/Barcode	DeviceID: 8F613DE3, BatchID: 37271A, ManufacturingDate: 12-02-2025, SerialNumber: 6FA9DE98D, Location: Plant C, Mysuru, RoHSCompliance: yes
21-06-2025 15:05		37271A	Passed	/content/drive/MyDrive/label_in	OCR	CrystalClear Springs . —e Natural Spring Water - 500 mL [a] 5 ae hl] . [a] : ts, 1 om Po) py te Cw d if +S Bottled by CrystalClear Inc., Pittsburgh, PA zi oe Pe ha: F 1
21-06-2025 15:05	E7DD120A	3F8A99	Passed	/content/drive/MyDrive/label_in	QR/Barcode	DeviceID: E7DD120A, BatchID: 3F8A99, ManufacturingDate: 08-03-2025, SerialNumber: B9E6E24F3, Location: Plant U, Kakinada, RoHSCompliance: yes
21-06-2025 15:05		3F8A99	Passed	/content/drive/MyDrive/label_in	OCR	CrystalClear Springs ox Natural Spring Water - 500 mL [a] a uf =F uA [2] AE ete belt ip os a d Bottled by CrystalClear Inc., Pittsburgh, PA bot . ote BP p Source: E
21-06-2025 15:05	80C7435B	6E0BFA	Passed	/content/drive/MyDrive/label_in	QR/Barcode	DeviceID: 80C7435B, BatchID: 6E0BFA, ManufacturingDate: 06-04-2025, SerialNumber: 6C2D0A0A3, Location: Plant B, Agra, RoHSCompliance: yes
21-06-2025 15:05		6E0BFA	Passed	/content/drive/MyDrive/label_in	OCR	CrystalClear Springs . 7 i ny oa Natural Spring Water - 500 mL | Er he at '[o] Coe a" i m Bish alae Bottled by CrystalClear Inc., Pittsburgh, PA i: a oS pot ok n,* Sou
21-06-2025 15:05	22F22873	55A404	Passed	/content/drive/MyDrive/label_in	QR/Barcode	DeviceID: 22F22873, BatchID: 55A404, ManufacturingDate: 28-02-2025, SerialNumber: 3CE8E5AF0, Location: Plant M, Nagpur, RoHSCompliance: yes
21-06-2025 15:05			Rejected	/content/drive/MyDrive/label_in	OCR	CrystalClear springs — Natural Spring Water - 500 mL | = ca [uo] Bucy TT usd» Petes a alee Bottled by CrystalClear Inc., Pittsburgh, PA ae x : n rea on Source: B
21-06-2025 15:05	DCB80AF9	69232	Passed	/content/drive/MyDrive/label_in	QR/Barcode	DeviceID: DCB80AF9, BatchID: 69232, ManufacturingDate: 01-03-2025, SerialNumber: F4FF254FC, Location: Plant S, Bhilai, RoHSCompliance: yes
21-06-2025 15:05		69232	Passed	/content/drive/MyDrive/label_in	OCR	CrystalClear Springs . wee Natural Spring Water - 500 mL [ole r 0 : il [5] 7H Boe te oy = Bottled by CrystalClear Inc., Pittsburgh, PA er = x. x : Source: Blue Ridge
21-06-2025 15:05	AD702784	621978	Passed	/content/drive/MyDrive/label_in	QR/Barcode	DeviceID: AD702784, BatchID: 621978, ManufacturingDate: 30-04-2025, SerialNumber: 7D5ECB81E, Location: Plant K, Gwalior, RoHSCompliance: yes
21-06-2025 15:05		621978	Passed	/content/drive/MyDrive/label_in	OCR	UrystalUlear oprings . = Natural Spring Water - 500 mL [o] rid rie Aer [ol] Bottled by CrystalClear Inc., Pittsburgh, PA ear at: Ane ; Source: Blue Ridge Springs, PA
21-06-2025 15:05	32B7F9C2	65AC13	Passed	/content/drive/MyDrive/label_in	QR/Barcode	DeviceID: 32B7F9C2, BatchID: 65AC13, ManufacturingDate: 09-03-2025, SerialNumber: 8B18B3EB8, Location: Plant Q, Indore, RoHSCompliance: yes
21-06-2025 15:05		65AC13	Passed	/content/drive/MyDrive/label_in	OCR	CrystalClear Springs samme Natural Spring Water - 500 mL [oj ie iapien 'o] ao ae . om er Bottled by CrystalClear Inc., Pittsburgh, PA ca bal. . aoe F Source; Blue
21-06-2025 15:05	1B9512FA	6C53DD	Passed	/content/drive/MyDrive/label_in	QR/Barcode	DeviceID: 1B9512FA, BatchID: 6C53DD, ManufacturingDate: 11-04-2025, SerialNumber: 5F577B0D4, Location: Plant I, Noida, RoHSCompliance: yes
21-06-2025 15:05		6C53DD	Passed	/content/drive/MyDrive/label_in	OCR	CrystalClear Springs - pre at oe Natural Spring Water - 500 mL [a] 1h Ty Oe [a] r en Jaen r Cars er) Bottled by CrystalClear Inc., Pittsburgh, PA Bs a A Lee Source: E

total_label(1)

7.4 Detect Defects Using CNN(ML model)

Anomaly Detection Model Development

This implementation step focuses on developing and training the core anomaly detection model using deep learning feature extraction combined with traditional machine learning algorithms. The system is designed to automatically identify defective products in manufacturing environments.

Technical Approach

1. **Feature Extraction Strategy**
    - Pre-trained CNN Models: Used pretrained ResNet model architectures (ResNet18, ResNet50, Wide-ResNet50-2) as Resnet works the best for robust feature extraction
    - Transfer Learning: Resnet is trained on ImageNe. Its pre-trained weights to extract meaningful visual features without the case of training the large labeled dataset.
    - Feature Dimensionality: Extracted 512-2048 dimensional feature vector. It works depending on the Resnet model. If the model is Resnet18, its value will be 512 and the other value for ResNet50, Wide-ResNet50-2.
2. **Anomaly Detection Methods**

In this project for the ML model we mainly used two complementary approaches:

a. Principal Component Analysis (PCA)

- It is unsupervised reconstruction-based anomaly detection method
- Faster training and inference
- In this method, the working performs in a compressed space. If the image can be constructed in the compressed space, it is defined as good, otherwise it will go into the anomaly section.

b. One-Class Support Vector Machine (OCSVM)

- Boundary-based anomaly detection. It draws a boundary around “normal”
- It trains on the normal data. So, it learns decision boundary encompassing normal samples
- Any image which is detected outside from the learning section is flagged i.e the anomaly

3. **Technical Implementation Stack:**
    - Deep Learning Libraries: Imported PyTorch with torchvision for CNN model loading and image preprocessing transformations.
    - Computer Vision Processing: Used PIL for the task of image handling and matplotlib for visualization of results and metrics of the data.
    - Machine Learning Frameworks: Imported Scikit-learn because it provides PCA dimensionality reduction and One-Class SVM anomaly detection algorithms.
    - Data Pipeline: Used the data pipeline of Custom PyTorch Dataset/DataLoader for better performance resulting in efficient batch processing of image datasets
    - Utilities: Imported NumPy for performing all the numerical operations, and pandas for data exportation, and joblib for model persistence for the whole process.

Implementation Architecture

The flow of project in this project follows as:

- We selected a dataset(Mvtech) which contains all the images of bottles, screws,cables,etc. We specifically worked with the product “water bottle” for our project. We selected 50 images of water bottles and added them into our drive. After adding them into our drive,we copied the paths of the selected images and assigned each image path for each device id in our data.
- In the working process of the model, first the raw image of the bottle will be the input. The image of the water bottle will be resized based on the ImageNet Stats and then the CNN feature Extraction(Resnet) will work on the image.
- After the work of CNN on the image, the anomaly detection is done using PCA+OCSVM, based on the results of scores which are resulted from the anomaly detection the final results take place in the form of visuals and the main data stored in excel sheet.

Comprehensive Dataset Analysis

Before taking our decision step, we first performed an initial analysis of different categories within the entire dataset:

- First, we explored all the available product categories within the dataset

- And then using different tech models we decided on the best performing model.
- We worked on cable category data within the dataset with Resnet model and achieved accuracy of 83 percent and mobilenet model with accuracy of 82 and wide_resnet50_2 with result of 77 percent accuracy.

Flexible Dataset Handling

- Used MVtech dataset from the datasets given along with the problem statement
- In the MVtech dataset, we mainly used the bottle dataset images. We focused on the images of the bottle, so we used the bad, good,etc of the bottle images which are available in MVtech dataset and added them into our google drive for better performance
- Robust label assignment based on the folder structure creation and added filename keywords
- Error handling for corrupted or unreadable images in our code.

Comprehensive Evaluation

- We used the most important metrics: AUROC, Average Precision, F1-Score, Accuracy
- Threshold optimization using statistical analysis and F1-maximization for better performance
- Per-image analysis with confidence scores and prediction details in the model

Results Export and Visualization

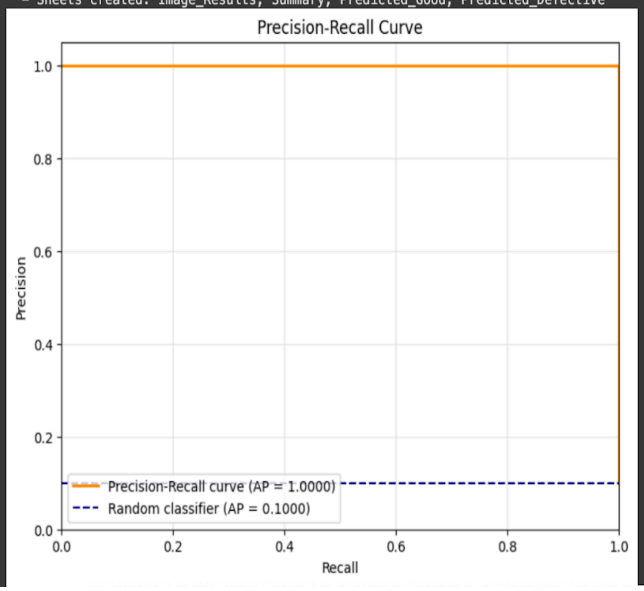
- As a result, we stored the results in Excel reports with detailed per-image results and summary statistics of the images of bottle
- Added visual feedback showing sample predictions with scores of the assigned images
- Performance curves (Precision-Recall, score distributions)

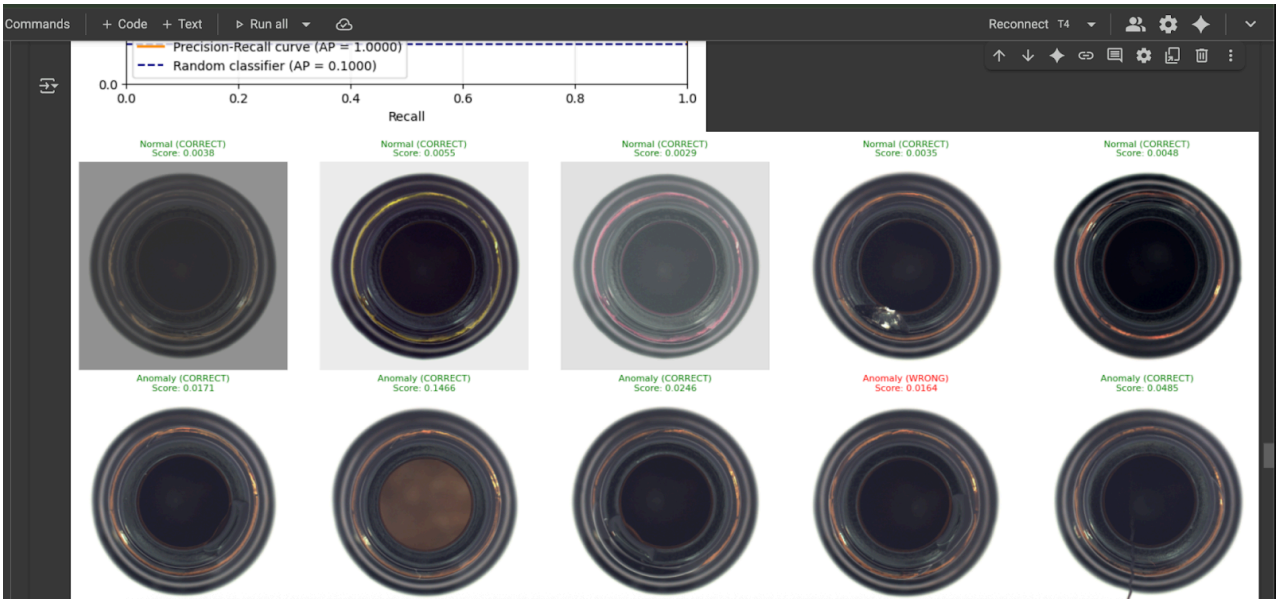
Commands + Code + Text ▷ Run all Reconnect T4

```
Using device: cuda
Downloading: "https://download.pytorch.org/models/resnet18-f37072fd.pth" to /root/.cache/torch/hub/checkpoints/resnet18-f37072fd.pth
100%|██████████| 44.7M/44.7M [00:00<00:00, 203MB/s]
Dataset type: flat
Found 50 images:
  Normal: 45
  Anomaly: 5

Extracting features...
Extracting features: 100%|██████████| 13/13 [00:45<00:00,  3.49s/it]
Extracted features shape: (50, 512)
Using 45 normal samples for training
Training PCA with 45 normal samples...
PCA components: 22
Explained variance ratio: 0.9521
Threshold set to: 0.005922

AUROC: 1.0000
Average Precision: 1.0000
Optimal threshold (best F1): 0.016429

Using threshold: 0.016429

Classification Report:
              precision    recall  f1-score   support

      Normal       0.98      1.00      0.99        45
     Anomaly       1.00      0.80      0.89         5

    accuracy                           0.98        50
   macro avg       0.99      0.90      0.94        50
weighted avg       0.98      0.98      0.98        50

Accuracy: 0.9800
```

Commands + Code + Text ▷ Run all Reconnect T4

```
- Total images processed: 50
- Results saved to: /content/results.xlsx
- Sheets created: Image_Results, Summary, Predicted_Good, Predicted_Defective
```





1. **Class-Specific Results**

```
AUROC: 1.0000
Average Precision: 1.0000
Optimal threshold (best F1): 0.016429

Using threshold: 0.016429

Classification Report:
              precision    recall  f1-score   support

      Normal       0.98      1.00      0.99        45
     Anomaly       1.00      0.80      0.89         5

    accuracy                           0.98        50
   macro avg       0.99      0.90      0.94        50
weighted avg       0.98      0.98      0.98        50

Accuracy: 0.9800
```

Key Insights

- Excellent detection capability with near-perfect metrics of the model
- No false positives for anomaly detection (100% precision)
- Missed 1 out of 5 anomalies (80% recall) resulting better predicting results
- PCA captured 95.2% variance with 22 components with MVtech dataset bottle images
- Fast processing: ~3.5 seconds per batch with GPU acceleration better than other models.

Performance Summary

1. *System Configuration*
   - Device: CUDA GPU acceleration for the ML model better performance
   - Dataset: 50 bottle images (45 normal, 5 anomaly) assigned to each device id, batch id.
   - Feature Extractor: ResNet18 pretrained model using learning of Imagenet
   - Method: PCA-based anomaly detection
2. *Model Performance*
   - AUROC(Area Under the Receiver Operating Characteristic Curve): 1.0000 (Perfect)
   - Average Precision: 1.0000 (Perfect)
   - Overall Accuracy: 98%

Overall Assessment

Good performance for anomaly detection of the dataset. The model results in strong capability to distinguish between normal and anomalous bottle images with minimal false alarms using the model.

Code Structure

Python

```
# Main components implemented:
class CustomDataset:          # Flexible dataset loading
class FeatureExtractor:       # CNN-based feature extraction
def train_pca_detector():     # PCA anomaly detection
def train_ocsvm_detector():   # OCSVM anomaly detection
def save_results_to_excel():  # Comprehensive reporting
def visualize_results():      # Visual result interpretation
```

Configuration Parameters

Python

```
DATASET_PATH = "/path/to/dataset"
FEATURE_EXTRACTOR = "resnet18"  # resnet18, resnet50, wide_resnet50_2
METHOD = "pca"                  # pca, ocsvm
```

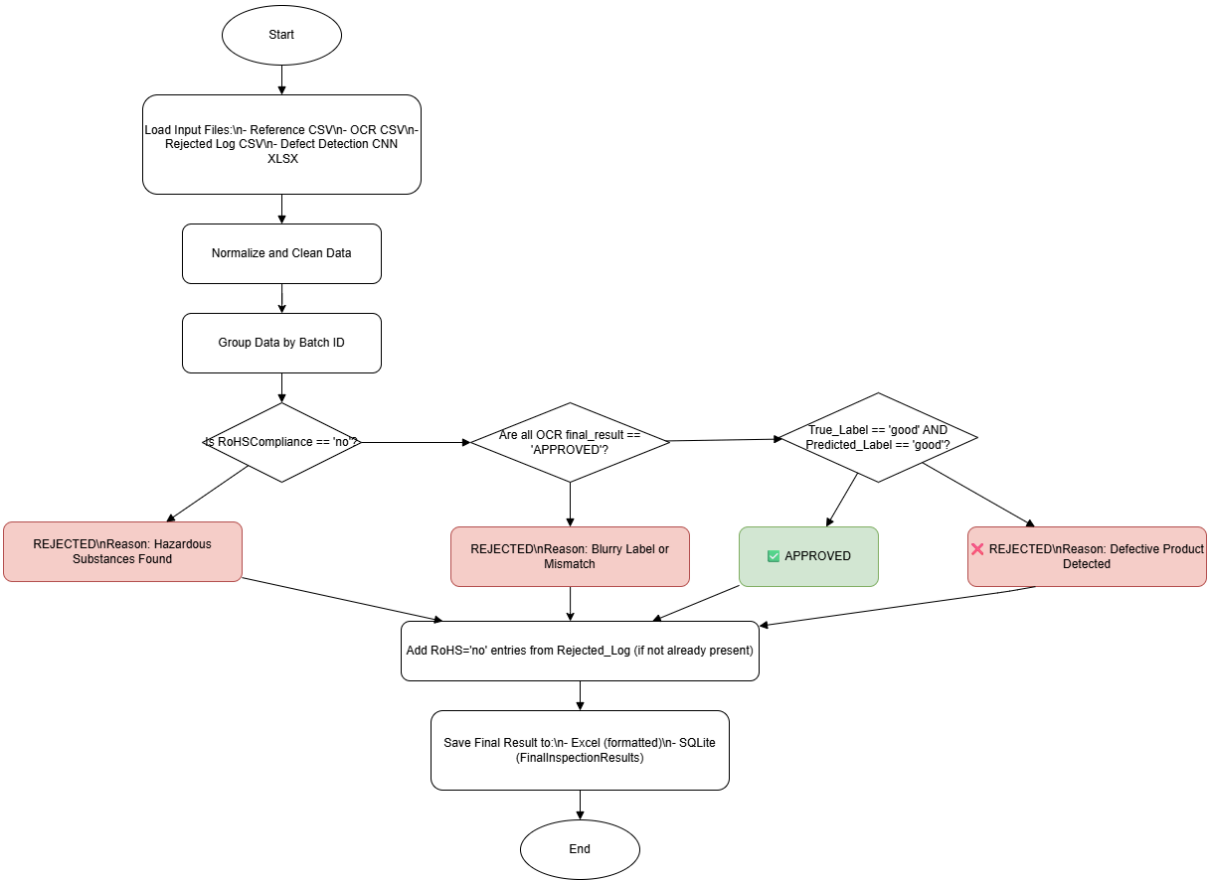
Other models performance results comparison

We tried other models like MobileNet which resulted with 82 percent accuracy, and Resnet50 which resulted in 77 percent. So, compared to all the other models, Resnet18 resulted 98 percent with highest accuracy.

Key Takeaway: ResNet18 is the best model comparatively, outperforming both (MobileNet) and (ResNet50) models. It performed with perfect AUROC/AP scores with minimal false positives.

This implementation step establishes the core anomaly detection capability that will be integrated into the complete quality control system in subsequent phases.

7.5 Final Inspection System



The final result includes the participation of the results of OCR, RoHS compliance, and defect detection models(using ML model). These results are validated and combined in this phase, the last phase of the project. For this final step, we used python for implementation,integrated OpenPyXL for Excel formatting, Pandas for data processing, and SQLite for structured traceability storage.

Technologies Used:

- Pandas: We used Pandas for tasks like data cleaning, data merging, data grouping, and logic handling.
- OpenPyXL: Used for Excel report generation tasks and color coding.

- SQLite3: Selected SQLite3 as it's easy to use and no server required for recording lightweight local databases which are used for inspection result logging.
- Datetime: Used for formatting the timestamp and fallback generation of the data.

1. File Integration and Input Loading

In the first step,the system first loads all the four main key input files:

- Reference CSV: Its main task is done by filling in the missing values by mapping the Batch IDs to Device IDs in the file.
- OCR Result CSV: It contains the final results of the OCR model. (e.g., APPROVED/REJECTED) and RoHS status of the bottles.
- Rejected Log CSV: It holds all the entries which are rejected due to RoHS S non-compliance of the bottle.
- Defect Detection ML model Result XLSX: It contains all the prediction results of the product images.

Dynamic file loading is handled using a helper function that accepts .csv or .xlsx formats.

2. Data Cleaning and Standardization

- In this step, we made column names standardized text fields
- We named the columns like RoHSCompliance, labels, and final results (e.g., correcting case).

All the missing Device IDs were filled automatically using a reference CSV file by matching Batch IDs, this step helps us to make sure that the data holds completeness and also simultaneously improves the analysis efficiency.

3. Batch-wise Inspection Logic

The data is first grouped by Batch ID and then evaluated with the following hierarchical logic using the data available :

- RoHS Check: If the result of RoHSCompliance == 'no', that particular batch is immediately marked as REJECTED with the reason "Hazardous Substances Found".
- OCR Result Check: If any of the final_result in the batch is not "APPROVED", it's marked as REJECTED due to "Blurry Label or Mismatch" as a defect is recognized.
- Model Validation: If all OCR results are approved, then the next step is validation of the batch, the validation of the batch is done by using AI model predictions. Only if both True_Label and Predicted_Label are resulted as "good", the product is APPROVED; otherwise, REJECTED for "Defective Product Detected".

4. Reinforcement with Rejected Log

In the results part, all the RoHS "no" entries from the Rejected Log are cross-checked if they are right or wrong. If any such batch was not part of the OCR evaluation from the initial step, it's added to the final result with a REJECTED status and with an appropriate reason.

5. Final Output Generation

For the final step, we generated two types of outputs:

a. Excel Report with Conditional Formatting

The first file is an excel .xlsx report generated using OpenPyXL.

In this result file, the rows are conditionally formatted:

- Green for APPROVED
- Red for REJECTED

This totally improves the readability and additionally provides quick visual

assessment .

b. SQLite Database Logging

For the database logging, we wrote the final DataFrame to a local SQLite database file named inspection_results.db.

All the final results are stored in a table format called FinalInspectionResults, which can be queried for multiple traceability purposes along with audit purposes.

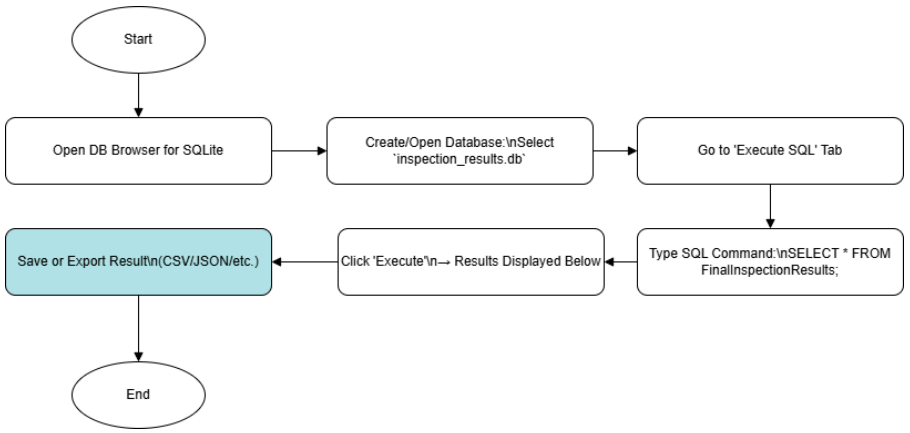
6. Preview and Verification

For a better verification step, we performed an optional preview of the top 5 entries which will be printed in the form the SQLite table, allowing users to verify that the data pipeline is functioning correctly end-to-end completely.

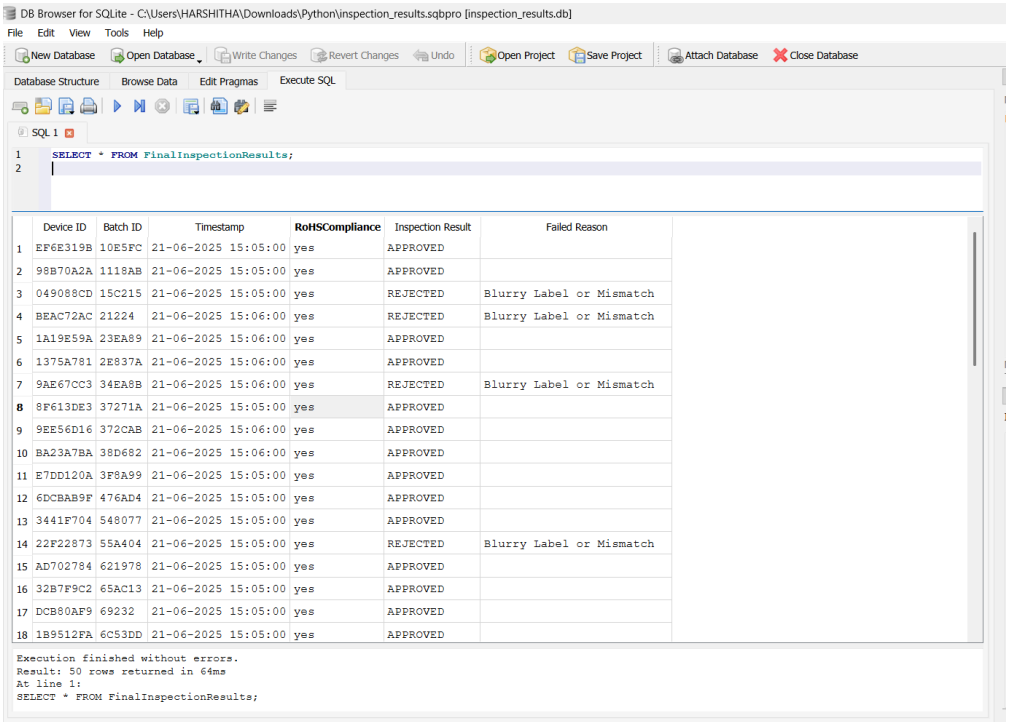
Final Output:

Output Type	**File Name**	**Description**
Excel	final_combined_result55.xlsx	Final results with color coding (green = approved, red = rejected)
Database	`inspection_results.db` (Table: `FinalInspectionResults`)	All inspection data stored in SQLite format for efficient querying

Flow in Database:



Results in Database:



	Device ID	Batch ID	Timestamp	RoHSCompliance	Inspection Result	Failed Reason
1	EF6E319B	10E5FC	21-06-2025 15:05:00	yes	APPROVED	
2	98B70A2A	1118AB	21-06-2025 15:05:00	yes	APPROVED	
3	049088CD	15C215	21-06-2025 15:05:00	yes	REJECTED	Blurry Label or Mismatch
4	BEAC72AC	21224	21-06-2025 15:06:00	yes	REJECTED	Blurry Label or Mismatch
5	1A19E59A	23EA89	21-06-2025 15:06:00	yes	APPROVED	
6	1375A781	2E837A	21-06-2025 15:06:00	yes	APPROVED	
7	9AE67CC3	34EA8B	21-06-2025 15:06:00	yes	REJECTED	Blurry Label or Mismatch
8	8F613DE3	37271A	21-06-2025 15:05:00	yes	APPROVED	
9	9EE56D16	372CAB	21-06-2025 15:06:00	yes	APPROVED	
10	BA23A7BA	38D682	21-06-2025 15:06:00	yes	APPROVED	
11	E7DD120A	3F8A99	21-06-2025 15:05:00	yes	APPROVED	
12	6DCBAB9F	476AD4	21-06-2025 15:05:00	yes	APPROVED	
13	3441F704	548077	21-06-2025 15:05:00	yes	APPROVED	
14	22F22873	55A404	21-06-2025 15:05:00	yes	REJECTED	Blurry Label or Mismatch
15	AD702784	621978	21-06-2025 15:05:00	yes	APPROVED	
16	32B7F9C2	65AC13	21-06-2025 15:05:00	yes	APPROVED	
17	DCB80AF9	69232	21-06-2025 15:05:00	yes	APPROVED	
18	1B9512FA	6C53DD	21-06-2025 15:05:00	yes	APPROVED	

Results in Excel sheet:

	A	B	C	D	E	F
1	Device ID	Batch ID	Timestamp	RoHSCompliance	Inspection Result	Failed Reason
2	EF6E319B	10E5FC	21-06-2025 15:05:00	yes	APPROVED	
3	98B70A2A	1118AB	21-06-2025 15:05:00	yes	APPROVED	
4	049088CD	15C215	21-06-2025 15:05:00	yes	REJECTED	Blurry Label or Mismatch
5	BEAC72AC	21224	21-06-2025 15:06:00	yes	REJECTED	Blurry Label or Mismatch
6	1A19E59A	23EA89	21-06-2025 15:06:00	yes	APPROVED	
7	1375A781	2E837A	21-06-2025 15:06:00	yes	APPROVED	
8	9AE67CC3	34EA8B	21-06-2025 15:06:00	yes	REJECTED	Blurry Label or Mismatch
9	8F613DE3	37271A	21-06-2025 15:05:00	yes	APPROVED	
10	9EE56D16	372CAB	21-06-2025 15:06:00	yes	APPROVED	
11	BA23A7BA	38D682	21-06-2025 15:06:00	yes	APPROVED	
12	E7DD120A	3F8A99	21-06-2025 15:05:00	yes	APPROVED	
13	6DCBAB9F	476AD4	21-06-2025 15:05:00	yes	APPROVED	
14	3441F704	548077	21-06-2025 15:05:00	yes	APPROVED	
15	22F22873	55A404	21-06-2025 15:05:00	yes	REJECTED	Blurry Label or Mismatch
16	AD702784	621978	21-06-2025 15:05:00	yes	APPROVED	
17	32B7F9C2	65AC13	21-06-2025 15:05:00	yes	APPROVED	
18	DCB80AF9	69232	21-06-2025 15:05:00	yes	APPROVED	
19	1B9512FA	6C53DD	21-06-2025 15:05:00	yes	APPROVED	
20	80C7435B	6E0BFA	21-06-2025 15:05:00	yes	APPROVED	
21	2FC18128	738105	21-06-2025 15:05:00	yes	APPROVED	
22	759DCBC3	7B9D13	21-06-2025 15:08:00	yes	APPROVED	
23	56D340A5	92E02C	21-06-2025 15:08:00	yes	REJECTED	Defective Product Detected
24	6075D2C5	96425E	21-06-2025 15:07:00	yes	APPROVED	
25	E7A9F12B	980035	21-06-2025 15:07:00	yes	APPROVED	

7.7 Batch ID Query & Result Viewer Interactive Streamlit UI (Extension Feature)

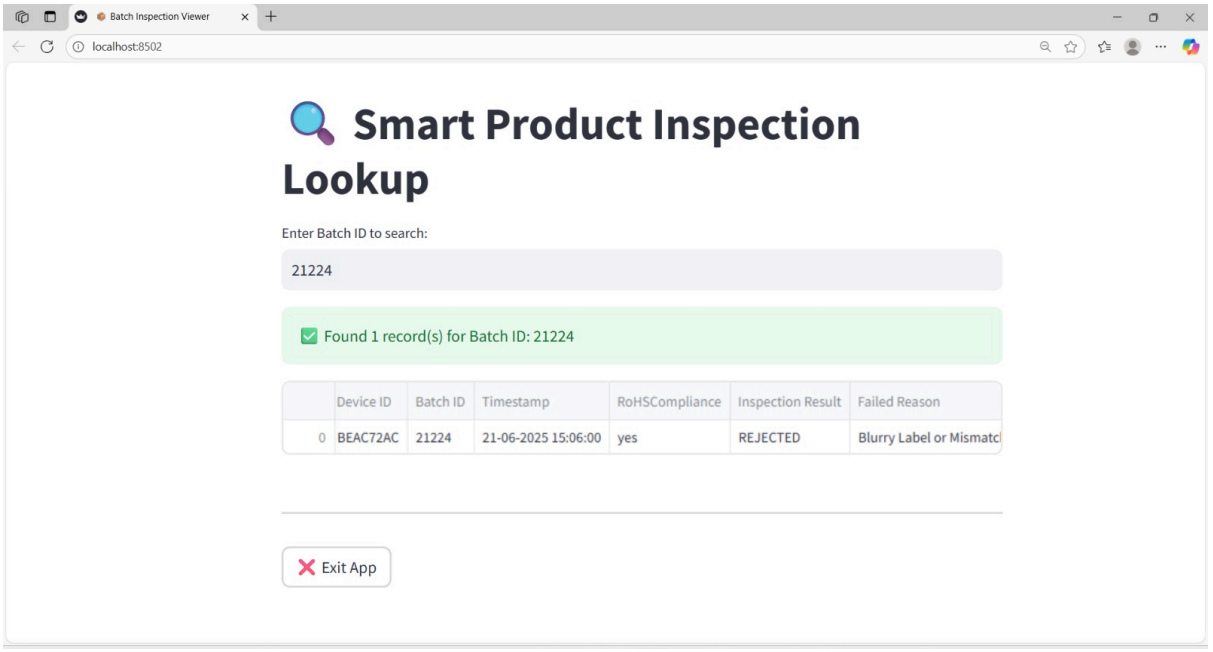
In the last step, we creed a User Interface(UI) in which the requested information of the product will be displayed as the output

This allows us to add the feature of real-time verification of product inspection records simply by using an interactive Streamlit-based UI.

Based upon the entered Batch ID of the product, it queries the details from the FinalInspectionResults table in the local SQLite database (inspection_results.db) and then displays the corresponding inspection details if found of the product.

Features:

- User input for Batch ID via web UI of the product
- SQL query is used to fetch the information of the product from the correct column mapping ([Batch ID])
- After fetching the information needed, then it displays full inspection record in a table
- Shutdown using Exit App button created that terminates the Streamlit process after completion of the task
- In this final step, we ensured that inspection results are easily accessible for traceability, review, and audit purposes and many other useful purposes during or after the quality control process of the product.



8. AI Module

For our anomaly detection system the core components of ML include:

Core ML Components:

- Feature extraction models (ResNet18, MobileNet, ResNet50)
- Dimensionality reduction (PCA) and in else OCSVM(One Class SVM)
- Anomaly detection algorithms
- Classification thresholds and scoring

Model Architecture:

- Pre-trained CNN backbones like Resnet18,etc
- Feature embedding pipeline(raw image to dimensional images that capture visual patterns)
- Anomaly scoring mechanism(using PCA/ OCSVM)

YOLOv5 Detection Model

i. YOLOv5 Model Training

To enable accurate localization and classification of product QR codes and text labels, a custom YOLOv5 model was trained and deployed using a complete pipeline covering dataset preparation, training, evaluation and export.

a. Dataset Annotation and Structure

- A custom dataset named **'label dataset'** was created with **198 images containing QR codes** and **248 images with printed product labels**.
- Manual annotation was performed using **LabelImg**, generating bounding boxes in **YOLO format** for:
    - **Class 0** – QR Code
    - **Class 1** – Product Label
- The data was split into **Train (70%)**, **Validation (20%)**, and **Test (10%)** sets using train_test_split.

Code Structure

```python
# Main components implemented:
```

```
def prepare_dataset():       # Organizes images/labels into YOLOv5 format
def create_data_yaml():      # Generates dataset config file for YOLO training
def train_yolov5_model():    # Trains YOLOv5 on QR + label images
def evaluate_model():        # Evaluates performance using classification metrics
def export_model_formats():  # Converts trained model to ONNX and TFLite formats
```

b. Custom YAML Configuration:

None

```
path: /content/datasets
train: images/train
val: images/val
test: images/test
nc: 2
names: ['qrcode', 'label']
```

c. Training Pipeline

The YOLOv5 pipeline was implemented using **Google Colab**.

d. Training Command:

Python

```
!python train.py --img 640 --batch 16 --epochs 50 \
--data data.yaml --weights yolov5s.pt --name
qr_label_detection
```

- The best model weights were saved as `best.pt`.

e.Model Evaluation (Test Set)

To assess performance, inference was run on the test dataset. Using actual vs predicted class labels, the following metrics was computed:

- **Classification Report** (Precision, Recall, F1)

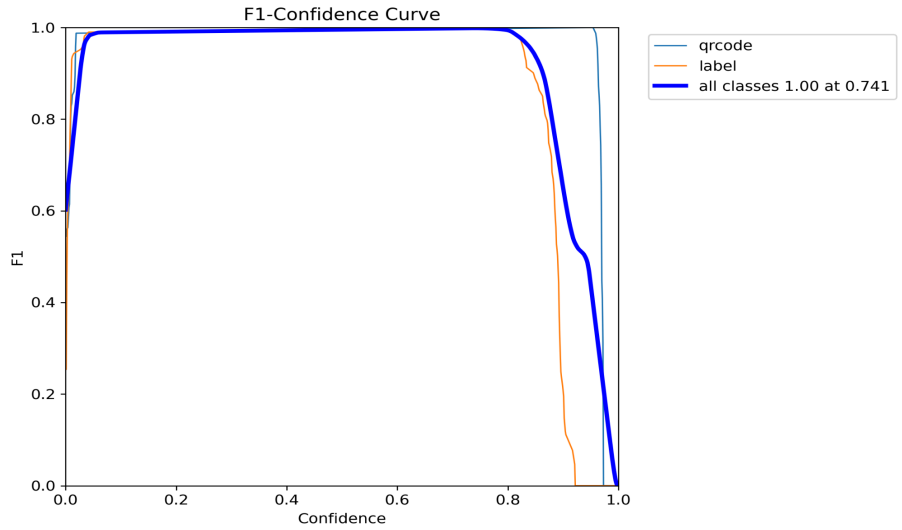
Class	Images	Instances	P	R	mAP50	mAP50-95: 100% 3/3 [00:02<00:00, 1.23it/s]
all	90	90	0.996	1	0.995	0.954
qrcode	90	40	0.994	1	0.995	0.934
label	90	50	0.999	1	0.995	0.973

This ensured quantitative assessment of detection accuracy for both QR and Label classes.

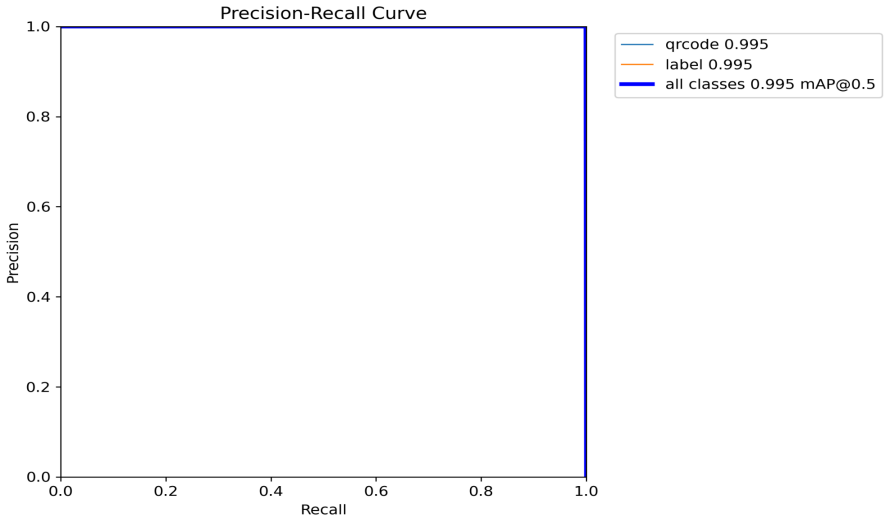
- **Performance Metrics and Results Report**

The performance metrics and model weights(best.pt, best.onnx, best.tflite) were saved to file path **'yolov5/runs/train/qr_label_detection'** in Google Colab files.

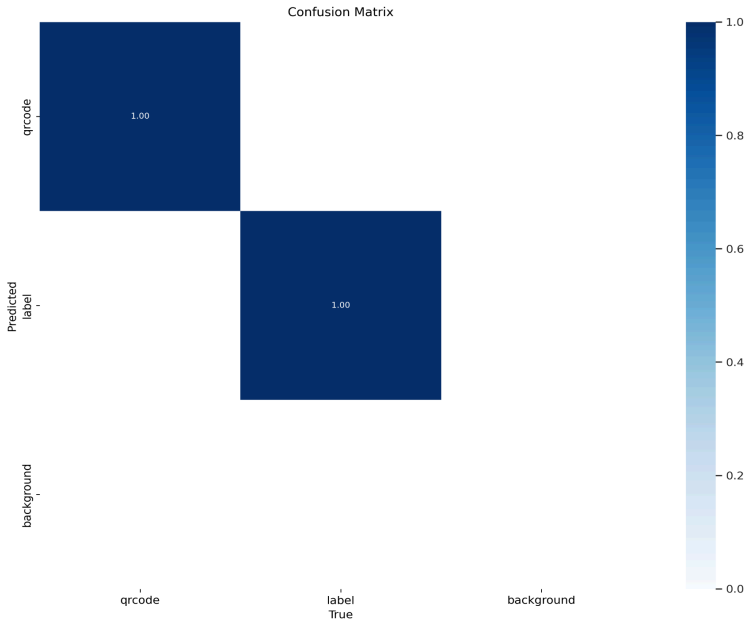
1. F1-Confidence Curve:



2.Precision-Recall Curve:



3.Confusion Matrix:

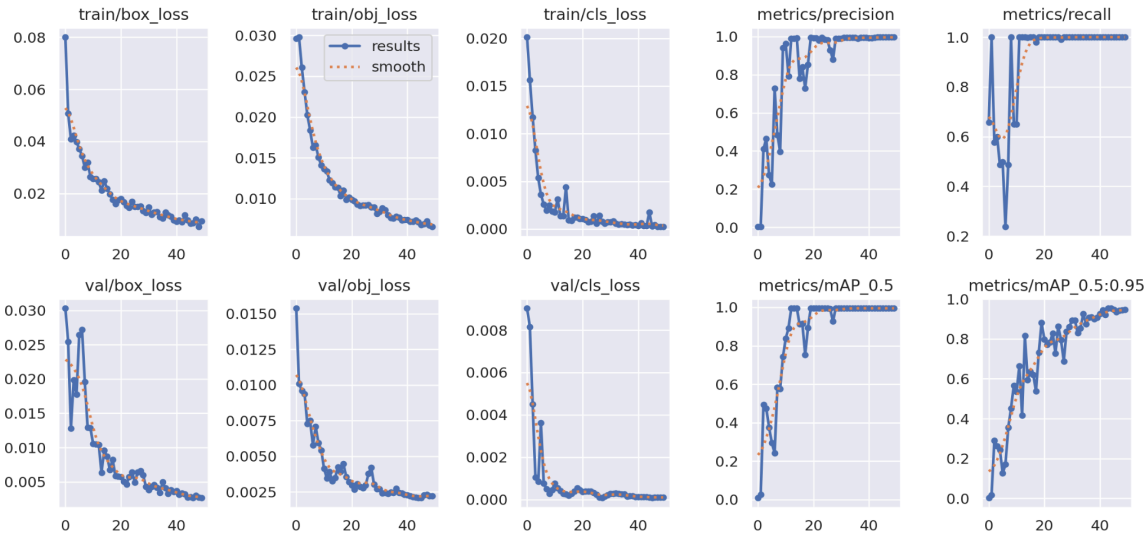


4.Results.csv:

epoch	train/box_loss	train/obj_loss	train/cls_loss	metrics/precision	metrics/recall	metrics/mAP_0.5	metrics/mAP_	val/box_loss	val/obj_loss	val/cls_loss	x/lr0	x/lr1	x/lr2
0	0.080047	0.029615	0.020126	0.0020922	0.6575	0.0088038	0.0025411	0.03033	0.015385	0.0090332	0.0829	0.0019	0.0019
1	0.050781	0.02978	0.015645	0.0033525	1	0.026604	0.017455	0.025418	0.010082	0.0081523	0.064823	0.0038228	0.0038228
2	0.040832	0.026069	0.011728	0.41179	0.5775	0.49473	0.2914	0.012768	0.0096084	0.0045151	0.046666	0.0056664	0.0056664
3	0.042302	0.023073	0.0082317	0.46545	0.6	0.47445	0.26205	0.019859	0.0093824	0.0010455	0.028431	0.0074307	0.0074307
4	0.039859	0.020258	0.0053675	0.27377	0.4875	0.37601	0.24319	0.017764	0.0072659	0.00084527	0.010116	0.0091159	0.0091159
5	0.037112	0.018361	0.0035916	0.22484	0.4975	0.29611	0.12615	0.026426	0.0075007	0.0036154	0.00901	0.00901	0.00901
6	0.034472	0.016297	0.0025726	0.72753	0.2375	0.24192	0.17053	0.027186	0.0057645	0.00076332	0.00901	0.00901	0.00901
7	0.029988	0.01657	0.0019233	0.48404	0.4875	0.58291	0.35478	0.019559	0.0070632	0.00050583	0.008812	0.008812	0.008812
8	0.031912	0.015074	0.0024759	0.39546	1	0.57575	0.45149	0.012901	0.0059496	0.00027847	0.008614	0.008614	0.008614
9	0.026425	0.014109	0.0018579	0.94034	0.65	0.74329	0.56583	0.01288	0.0054199	0.00046921	0.008416	0.008416	0.008416
10	0.025624	0.01362	0.0017507	0.96381	0.65	0.83731	0.53557	0.010534	0.0041361	0.00077254	0.008218	0.008218	0.008218
11	0.025762	0.01334	0.0031259	0.79166	1	0.89175	0.66343	0.010488	0.0034538	0.0005544	0.00802	0.00802	0.00802
12	0.024431	0.012296	0.0013502	0.98921	1	0.995	0.41715	0.010382	0.0039133	0.00040862	0.007822	0.007822	0.007822
13	0.021278	0.011915	0.0013388	0.98978	1	0.995	0.81506	0.0063475	0.0032444	0.00029411	0.007624	0.007624	0.007624
14	0.024762	0.011449	0.0043725	0.99317	0.99856	0.995	0.59443	0.0095877	0.0034961	0.00025393	0.007426	0.007426	0.007426
15	0.021998	0.011385	0.00090996	0.78048	1	0.9117	0.63596	0.0086906	0.0042424	0.0001948	0.007228	0.007228	0.007228
16	0.019769	0.010319	0.00088194	0.84009	1	0.91518	0.6206	0.0067708	0.003893	0.0002751	0.00703	0.00703	0.00703
17	0.017727	0.011048	0.001191	0.72773	0.98	0.75235	0.53824	0.0082658	0.004473	0.00040461	0.006832	0.006832	0.006832
18	0.016049	0.0099073	0.001227	0.85325	1	0.89381	0.73142	0.0058572	0.0035533	0.00054289	0.006634	0.006634	0.006634
19	0.017447	0.010186	0.0010786	0.99336	1	0.995	0.88229	0.0057592	0.0031793	0.00045827	0.006436	0.006436	0.006436
20	0.018017	0.0098992	0.0010773	0.99157	1	0.995	0.79755	0.0057029	0.0029432	0.00036182	0.006238	0.006238	0.006238
21	0.016684	0.0097098	0.00095685	0.99209	1	0.995	0.77248	0.005054	0.0026703	0.00038427	0.00604	0.00604	0.00604
22	0.015232	0.0092956	0.0007329	0.98007	1	0.99441	0.78294	0.0046272	0.0030472	0.00038433	0.005842	0.005842	0.005842
23	0.014546	0.009166	0.00076381	0.99377	1	0.995	0.82747	0.0057379	0.0028737	0.00040042	0.005644	0.005644	0.005644

results (1)

5.Training and Validation Metrics:



- Top row depicts training metrics and bottom row depicts validation metrics.
- **Training Metrics:**

  train/box_loss, train/obj_loss, train/cls_loss, metrics/precision, metrics/recall

- **Validation Metrics:**

  val/box_loss, val/obj_loss, val/cls_loss, metrics/mAP_0.5, metrics/mAP_0.5:0.95

f. Model Export

To support lightweight deployment, the trained model was exported to **ONNX Format** as **best.onnx**

Command:

```python
!python export.py --weights best.pt --include onnx --img 640
```

YOLOv5 Model Architecture:

```
Overriding model.yaml nc=80 with nc=2

                 from  n    params  module                                  arguments
  0                -1  1      3520  models.common.Conv                      [3, 32, 6, 2, 2]
  1                -1  1     18560  models.common.Conv                      [32, 64, 3, 2]
  2                -1  1     18816  models.common.C3                        [64, 64, 1]
  3                -1  1     73984  models.common.Conv                      [64, 128, 3, 2]
  4                -1  2    115712  models.common.C3                        [128, 128, 2]
  5                -1  1    295424  models.common.Conv                      [128, 256, 3, 2]
  6                -1  3    625152  models.common.C3                        [256, 256, 3]
  7                -1  1   1180672  models.common.Conv                      [256, 512, 3, 2]
  8                -1  1   1182720  models.common.C3                        [512, 512, 1]
  9                -1  1    656896  models.common.SPPF                      [512, 512, 5]
 10                -1  1    131584  models.common.Conv                      [512, 256, 1, 1]
 11                -1  1         0  torch.nn.modules.upsampling.Upsample    [None, 2, 'nearest']
 12          [-1, 6]  1         0  models.common.Concat                    [1]
 13                -1  1    361984  models.common.C3                        [512, 256, 1, False]
 14                -1  1     33024  models.common.Conv                      [256, 128, 1, 1]
 15                -1  1         0  torch.nn.modules.upsampling.Upsample    [None, 2, 'nearest']
 16          [-1, 4]  1         0  models.common.Concat                    [1]
 17                -1  1     90880  models.common.C3                        [256, 128, 1, False]
 18                -1  1    147712  models.common.Conv                      [128, 128, 3, 2]
 19         [-1, 14]  1         0  models.common.Concat                    [1]
 20                -1  1    296448  models.common.C3                        [256, 256, 1, False]
 21                -1  1    590336  models.common.Conv                      [256, 256, 3, 2]
 22         [-1, 10]  1         0  models.common.Concat                    [1]
 23                -1  1   1182720  models.common.C3                        [512, 512, 1, False]
 24      [17, 20, 23]  1     18879  models.yolo.Detect                      [2, [[10, 13, 16, 30, 33, 23], [30, 61, 62, 45, 59, 119], [116, 90, 156, 198, 373, 326]], [128, 256, 512]]
Model summary: 214 layers, 7025023 parameters, 7025023 gradients, 16.0 GFLOPs
```

- Number of layers: **157**
- Number of parameters: **70,15,519**
- Number of epochs: **50**

g. Libraries & Tools Used

- Object Detection-> YOLOv5 (Ultralytics, PyTorch)
- Image Annotation-> LabelImg
- Data Handling & Split-> pandas, sklearn
- QR Decoding-> pyzbar
- Export-> ONNX

ii. Loading YOLOv5 Custom Trained Model for Label Detection

In order to extend the capabilities of the YOLOv5 object detection system, a comprehensive Python module was built to:

- Perform real-time label detection using a webcam or image input
- Apply OCR using Tesseract to extract textual data from detected label regions
- Decode a wide range of metadata formats including QR/barcodes, base64, hex, URLs, JSON, Morse code, Caesar cipher, and more

Pipeline Capabilities

Module	Functionality
YOLOv5 Detector	Detects text regions and QR codes in video/image frames
Tesseract OCR	Extracts text from detected label regions

Pyzbar	Decodes QR codes and barcodes
Regex Decoding Engine	Recognizes and decodes encoded formats (e.g., base64, hex, URL)
Text Analysis Engine	Calculates entropy, detects language hints, and estimates readability
Result Annotation & Display	Overlays decoded metadata and bounding boxes on live video or static images

Modes of Operation

1. Webcam Mode

- Real-time video feed processed frame-by-frame
- Detected labels and QR codes are highlighted live
- Decoded info is displayed on screen and printed to console
- User can press:
  - `q` to quit
  - `s` to save the current frame
  - `r` to export the current decoded data
  - `d` to toggle detailed debug output

2. Single Image Mode

- Static image is processed
- All detected metadata is drawn on the image
- Output is saved both as:
  - Annotated image (`.jpg`)
  - Decoded metadata (`.json`)
- Supports batch testing of dataset images

Command-Line Usage

1. Webcam Mode

```
None
python main.py --model best.pt --mode webcam
```

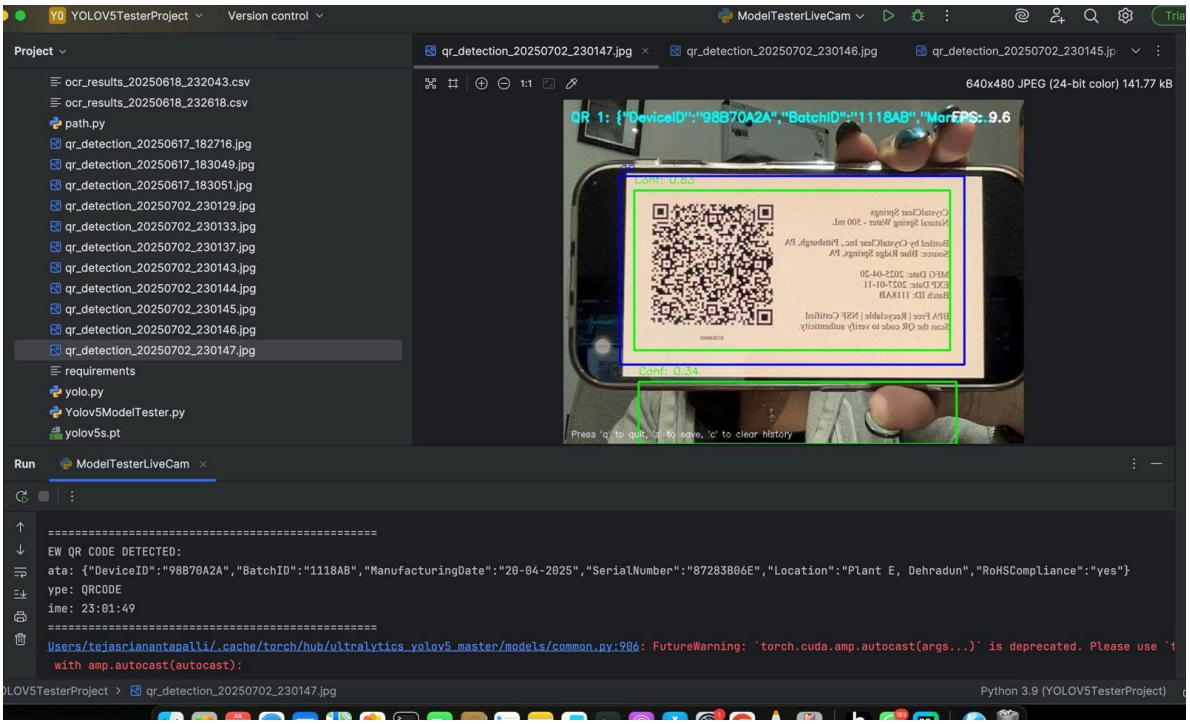
2. Image Mode

None

```
python main.py --model best.pt --mode image --image
path/to/image.jpg
```

Output

- Annotated Frame with bounding boxes, OCR confidence, decoded labels



Significance

This module enables:

- Rapid real-time inspection of product labels in industrial setups
- Flexible integration with multiple input types (video/image)
- Automated detection of tampered or misleading metadata
- Debuggable and extensible architecture for further enhancement

9. Traceability Logging

Traceability Logging is the last step in any intelligent inspection system, processing each product batch's inspection history. This supports audits and quality inspections while providing transparency and accountability and aids in tracking faulty or non-conforming products

9.1 Log Fields

The following fields are captured and saved in both the **Excel report** and the **SQLite database** (`inspection_results.db`, table: `FinalInspectionResults`):

Field Name	Description
Device ID	Unique identifier of the product
Batch ID	Identifier representing the product batch
Timestamp	Date and time of inspection
RoHSCompliance	Indicates whether the product is compliant with RoHS (yes/no)
Inspection Result	Final decision after inspection: `APPROVED` or `REJECTED`
Failed Reason	If rejected, specifies the reason (e.g., *Hazardous Substances Found*, *Blurry Label*, *Defective Product*)

These fields ensure both **traceability** and **explainability** of the inspection outcomes.

9.2 Inspection Results

The `Inspection Result` field is determined using a **three-tiered decision logic**, ensuring a comprehensive quality evaluation.

Step	Category	Criteria	Result	Reasons for Rejection
1.	RoHS Compliance	If RoHSCompliance == "no"	REJECTED	Hazardous Substances Found
2.	OCR Label Verification	If **any** final_result != "APPROVED" in the batch	REJECTED	Blurry Label or Mismatch
3.	Defect Detection AI Model Validation	If True_Label == "good" AND Predicted_Label == "good" Else (either label is not good)	APPROVED REJECTED	Empty Defective Product Detected

9.3 Tools Used

Tool / Library	Purpose
Pandas	For dataset processing, merging, filtering, and group-wise analysis
OpenPyXL	For writing formatted Excel output with color coding
SQLite3	For storing structured inspection data for traceability queries
datetime	For timestamp formatting and automatic fallback generation
os	For handling file paths and extensions dynamically

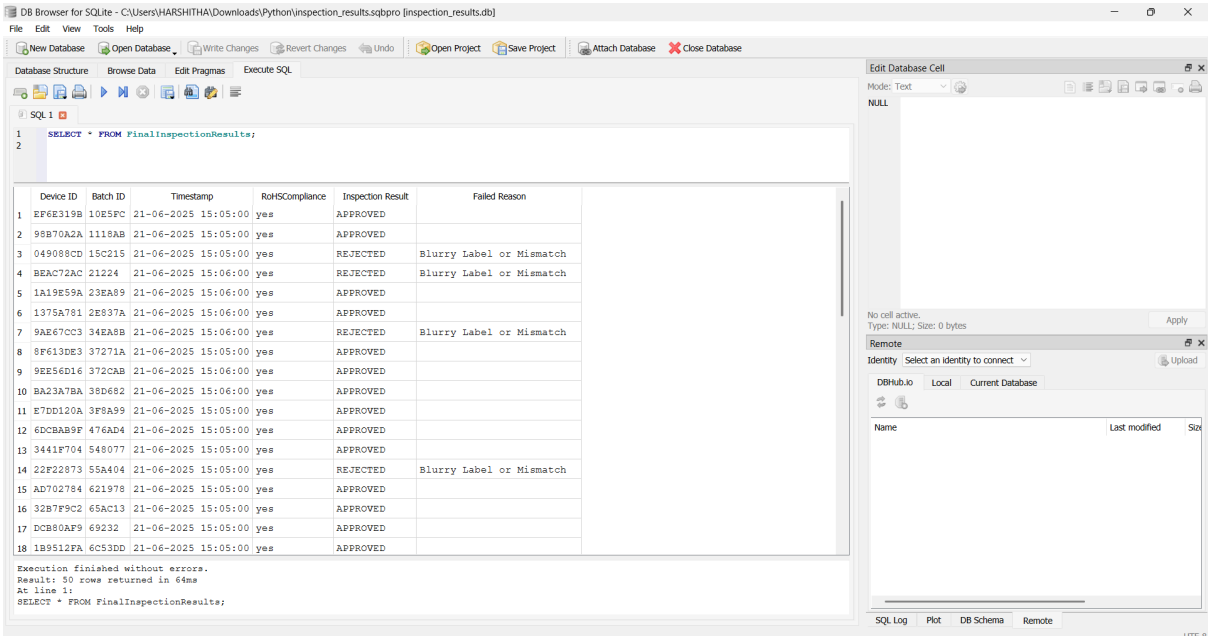
9.4 Output Files Generated

Output Type	File Name	Description
Excel	final_combined_result55.xlsx	Final results with color coding (green = approved, red = rejected)
Database	`inspection_results.db` (Table: `FinalInspectionResults`)	All inspection data stored in SQLite format for efficient querying

FLOW SQLDataBase:

DB Browser for SQLite —> New Database —> Select file inspection_results —> Execute SQL—> Command (SELECT * FROM FinalInspectionResults;) —> Result Displayed —> Save or Export Results.

Sample outputs shown below (Database and Excel) :

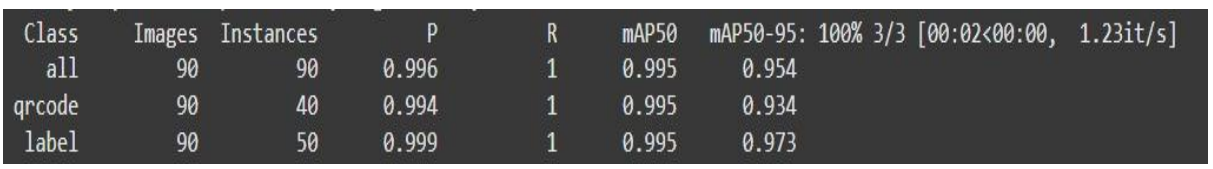


	Device ID	Batch ID	Timestamp	RoHSCompliance	Inspection Result	Failed Reason
1	EF6E319B	10E5FC	21-06-2025 15:05:00	yes	APPROVED	
2	98B70A2A	1118AB	21-06-2025 15:05:00	yes	APPROVED	
3	049088CD	15C215	21-06-2025 15:05:00	yes	REJECTED	Blurry Label or Mismatch
4	BEAC72AC	21224	21-06-2025 15:06:00	yes	REJECTED	Blurry Label or Mismatch
5	1A19E59A	23EA89	21-06-2025 15:06:00	yes	APPROVED	
6	1375A781	2E837A	21-06-2025 15:06:00	yes	APPROVED	
7	9AE67CC3	34EA8B	21-06-2025 15:06:00	yes	REJECTED	Blurry Label or Mismatch
8	8F613DE3	37271A	21-06-2025 15:05:00	yes	APPROVED	
9	9EE56D16	372CAB	21-06-2025 15:06:00	yes	APPROVED	
10	BA23A7BA	38D682	21-06-2025 15:06:00	yes	APPROVED	
11	E7DD120A	3F8A99	21-06-2025 15:05:00	yes	APPROVED	
12	6DCBAB9F	476AD4	21-06-2025 15:05:00	yes	APPROVED	
13	3441F704	548077	21-06-2025 15:05:00	yes	APPROVED	
14	22F22873	55A404	21-06-2025 15:05:00	yes	REJECTED	Blurry Label or Mismatch
15	AD702784	621978	21-06-2025 15:05:00	yes	APPROVED	
16	32B7F9C2	65AC13	21-06-2025 15:05:00	yes	APPROVED	
17	DCB80AF9	69232	21-06-2025 15:05:00	yes	APPROVED	
18	1B9512FA	6C53DD	21-06-2025 15:05:00	yes	APPROVED	
19	80C7435B	6E0BFA	21-06-2025 15:05:00	yes	APPROVED	
20	2FC18128	738105	21-06-2025 15:05:00	yes	APPROVED	
21	759DCBC3	7B9D13	21-06-2025 15:08:00	yes	APPROVED	
22	56D340A5	92E02C	21-06-2025 15:08:00	yes	REJECTED	Defective Product Detected
23	6075D2C5	96425E	21-06-2025 15:07:00	yes	APPROVED	
24	E7A9F12B	980035	21-06-2025 15:07:00	yes	APPROVED	

10. Results and Performance

10.1 Accuracy Metrics and Screenshots

a.YOLOv5 Model Performance Metrics:



The model correctly identified and classified all objects without any false negatives. Its precision was 0.996, and its recall was 1.0. The mAP was 0.995 at IoU 0.5, while mAP@50-95 was 0.954, indicating accurate object detection. Class label performance marginally

exceeded qrcode with a score of 0.973 versus 0.934. This shows how consistently and dependably it performed in both classes.

b. ML model result:

```
Classification Report:
              precision    recall  f1-score   support

      Normal       0.98      1.00      0.99        45
     Anomaly       1.00      0.80      0.89         5

    accuracy                           0.98        50
   macro avg       0.99      0.90      0.94        50
weighted avg       0.98      0.98      0.98        50

Accuracy: 0.9800
```

Focused on the accuracy for the results, using the Resnet model we got 90 percent accuracy along with the best scores of other metrics(precision, recall, f1 score and support).

Stored the results of all individual bottles in an excel file focusing on:

1. Result of the images:

	A	B	C	D	E	F	G	H	I	J	K
1	Image_ID	age_Filena	Image_Path	Directory	Anomaly_Score	Threshold	True_Label	Predicted_Label	Prediction_Status	Confidence	Above_Threshold
2	1	000.png	/content/drive/My	/content/drive/M	0.003724	0.016429	Good	Good	Correct	0.012706	FALSE
3	2	001.png	/content/drive/My	/content/drive/M	0.004737	0.016429	Good	Good	Correct	0.011693	FALSE
4	3	001_aug05	/content/drive/My	/content/drive/M	0.001973	0.016429	Good	Good	Correct	0.014456	FALSE
5	4	001_aug15	/content/drive/My	/content/drive/M	0.003989	0.016429	Good	Good	Correct	0.01244	FALSE
6	5	001_aug16	/content/drive/My	/content/drive/M	0.00375	0.016429	Good	Good	Correct	0.012679	FALSE
7	6	002.png	/content/drive/My	/content/drive/M	0.005455	0.016429	Good	Good	Correct	0.010974	FALSE
8	7	002_aug17	/content/drive/My	/content/drive/M	0.000869	0.016429	Good	Good	Correct	0.01556	FALSE
9	8	003.png	/content/drive/My	/content/drive/M	0.005123	0.016429	Good	Good	Correct	0.011306	FALSE
10	9	004.png	/content/drive/My	/content/drive/M	0.004887	0.016429	Good	Good	Correct	0.011542	FALSE
11	10	005.png	/content/drive/My	/content/drive/M	0.003715	0.016429	Good	Good	Correct	0.012714	FALSE

This is the format of the results in the excel file. All the images are predicted based on the threshold score, based on the threshold score the images will be resulted as either true or false in the data.

2. Summary of the result:

Metric	Value
Total Images	50
Good Images (True)	45
Defective Images (True)	5
Good Images (Predicted)	46
Defective Images (Predicted)	4
Correct Predictions	49
Incorrect Predictions	1
Accuracy	0.9800
Threshold Used	0.016429
Average Anomaly Score (Good)	0.003805
Average Anomaly Score (Defective)	0.050636

We worked on many images, within those we selected images which are of decent usage and then worked on the selected pictures picked from the Mvtech(bottle) dataset. The summary of the result holds that all the 50 images of the bottles are processed and tested, in that the good images are forty-five and bad ones with defects are five from the total.Here the predicted and the actual images result is compared, when compared all the 49 images are predicted correct and only on prediction went wrong. So, the total accuracy results with 98 percent. The whole prediction is done based on the constant threshold value along with the anomaly scores.

Results of Predicted_good images data:

Image_ID	Image_Filename	mage_Path	Directory	omaly_Sco	Threshold	True_Label	Predicted_Label	Prediction_Status	Confidence	Above_Threshold
1	000.png	/content/dr	/content/dr	0.003724	0.016429	Good	Good	Correct	0.012706	FALSE
2	001.png	/content/dr	/content/dr	0.004737	0.016429	Good	Good	Correct	0.011693	FALSE
3	001_aug05.png	/content/dr	/content/dr	0.001973	0.016429	Good	Good	Correct	0.014456	FALSE
4	001_aug15.png	/content/dr	/content/dr	0.003989	0.016429	Good	Good	Correct	0.01244	FALSE
5	001_aug16.png	/content/dr	/content/dr	0.00375	0.016429	Good	Good	Correct	0.012679	FALSE
6	002.png	/content/dr	/content/dr	0.005455	0.016429	Good	Good	Correct	0.010974	FALSE
7	002_aug17.png	/content/dr	/content/dr	0.000869	0.016429	Good	Good	Correct	0.01556	FALSE
8	003.png	/content/dr	/content/dr	0.005123	0.016429	Good	Good	Correct	0.011306	FALSE
9	004.png	/content/dr	/content/dr	0.004887	0.016429	Good	Good	Correct	0.011542	FALSE
10	005.png	/content/dr	/content/dr	0.003715	0.016429	Good	Good	Correct	0.012714	FALSE
11	005_aug04.png	/content/dr	/content/dr	0.004138	0.016429	Good	Good	Correct	0.012291	FALSE
12	005_aug09.png	/content/dr	/content/dr	0.00422	0.016429	Good	Good	Correct	0.012209	FALSE
13	006.png	/content/dr	/content/dr	0.001787	0.016429	Good	Good	Correct	0.014642	FALSE
14	006_aug18.png	/content/dr	/content/dr	0.002489	0.016429	Good	Good	Correct	0.01394	FALSE
15	006_aug19.png	/content/dr	/content/dr	0.001952	0.016429	Good	Good	Correct	0.014477	FALSE
16	007.png	/content/dr	/content/dr	0.005062	0.016429	Good	Good	Correct	0.011367	FALSE
17	007_aug01.png	/content/dr	/content/dr	0.006511	0.016429	Good	Good	Correct	0.009918	FALSE
18	007_aug02.png	/content/dr	/content/dr	0.002823	0.016429	Good	Good	Correct	0.013606	FALSE
19	008.png	/content/dr	/content/dr	0.006024	0.016429	Good	Good	Correct	0.010406	FALSE
20	008_aug07.png	/content/dr	/content/dr	0.002386	0.016429	Good	Good	Correct	0.014043	FALSE
21	009.png	/content/dr	/content/dr	0.005188	0.016429	Good	Good	Correct	0.011241	FALSE
22	010.png	/content/dr	/content/dr	0.003567	0.016429	Good	Good	Correct	0.012862	FALSE
23	010_aug06.png	/content/dr	/content/dr	0.001741	0.016429	Good	Good	Correct	0.014688	FALSE
24	011.png	/content/dr	/content/dr	0.004026	0.016429	Good	Good	Correct	0.012403	FALSE
25	012.png	/content/dr	/content/dr	0.003947	0.016429	Good	Good	Correct	0.012482	FALSE
26	012_aug08.png	/content/dr	/content/dr	0.005514	0.016429	Good	Good	Correct	0.010916	FALSE

Image_Results | Summary | Predicted_Good | Predicted_Defective

All the 45 predicted_good images are stored in the results excel sheet. For all the good ones the threshold score is good and the prediction is correct with a decent score of confidence. None of the predicted_good images threshold value is above the constant threshold value.

Results of Predicted_Defective images data:

A	B	C	D	E	F	G	H	I	J	K
Image_ID	age_Filena	mage_Path	Directory	Anomaly_Score	Threshold	True_Label	Predicted_Label	Prediction_Status	Confidence	Above_Threshold
47	defect_con	/content/dr	/content/dr	0.024611	0.016429	Defective	Defective	Correct	0.008181	TRUE
48	defect_con	/content/dr	/content/dr	0.146570995	0.016429	Defective	Defective	Correct	0.130142	TRUE
49	defect_con	/content/dr	/content/dr	0.048455	0.016429	Defective	Defective	Correct	0.032026	TRUE
50	defect_con	/content/dr	/content/dr	0.017115001	0.016429	Defective	Defective	Correct	0.000686	TRUE

Image_Results | Summary | Predicted_Good | Predicted_Defective | +

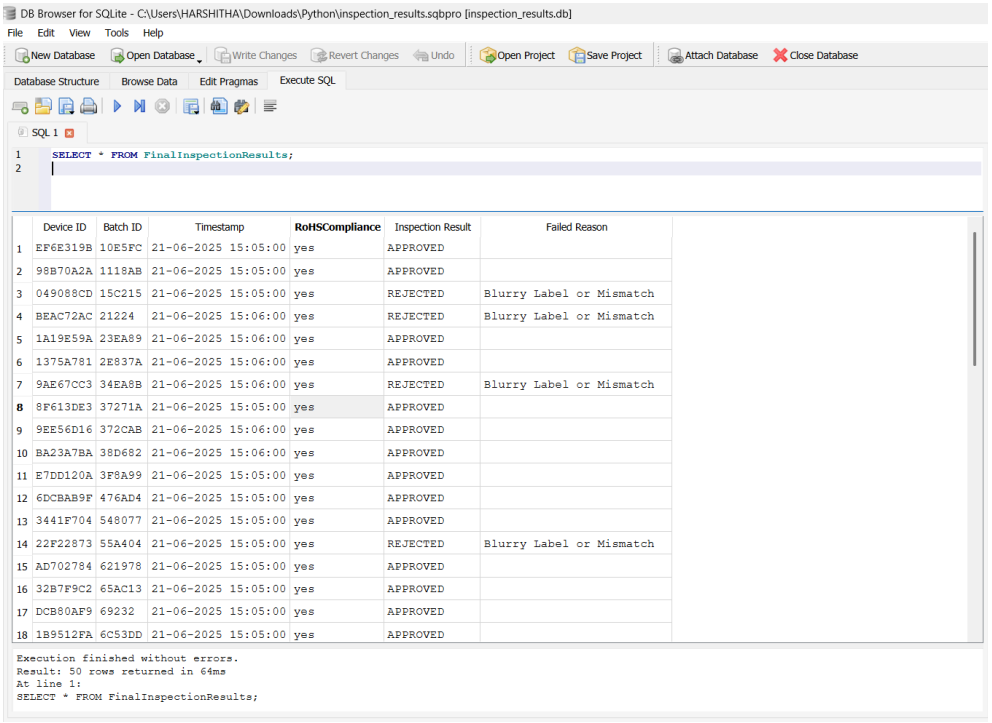
This is the result of all the defective images with value above the threshold with a decent score of confidence.The anomaly score is very high for all the defective images resulting in defective images.

All the results of each product image will be used in the next step for final results.

c. Traceability Logging

After accuracy evaluation and model validation, all final inspection outcomes — including decisions based on RoHS compliance, OCR label verification, and defect detection — are systematically captured in the traceability logging pipeline.

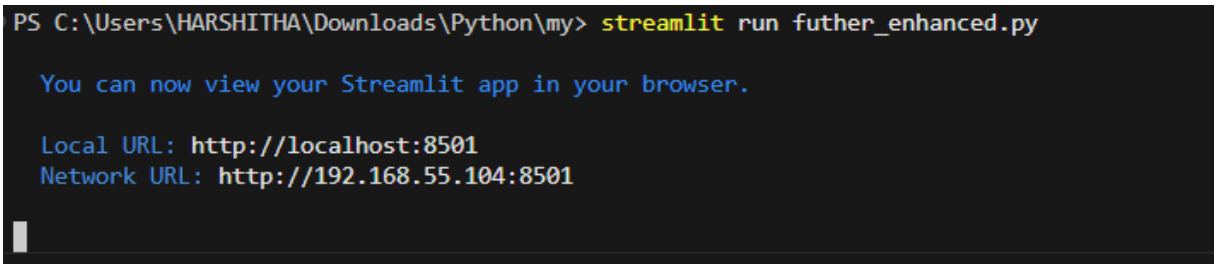
These results are stored in both Excel (`final_combined_result55.xlsx`) and SQLite database (`inspection_results.db`) formats, as detailed in **Section 9: Traceability Logging**.

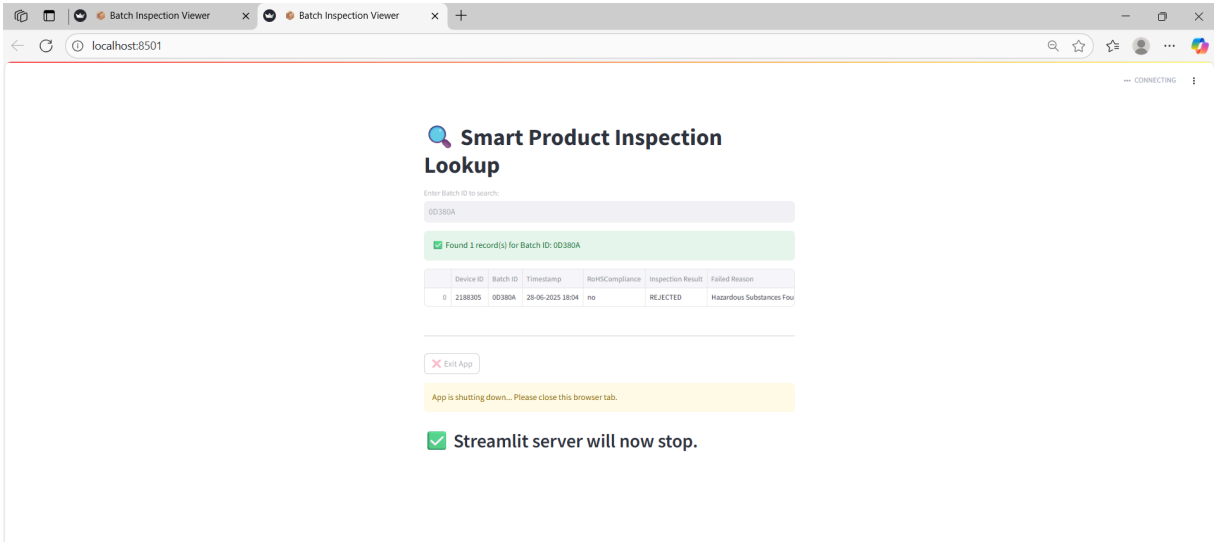


Batch ID Query & Result Viewer Streamlit UI (Extension Feature)

An interactive module built with Streamlit for real-time lookup of inspection results. Users enter a Batch ID to view detailed records from the `FinalInspectionResults` table in `inspection_results.db`.

It supports quick traceability, audits, and includes a clean shutdown via an **Exit App** button.





10.2 Data Definitions

Every data field utilised by the Smart Product Traceability and Inspection System is explained in this section. These data components are crucial for monitoring, evaluating, and confirming each product's compliance and quality. The structure and function of each piece of data collected during label verification, defect detection, and final decision-making are described in the definitions that follow.

Product_ID: Each item gets a special alphanumeric code so we can track it clearly in the database.

Batch_ID: Scanned straight from the products QR/Barcode, this tag tells us which production run the item came from.

RoHS_Compliance: Simple yes-or-no box showing if the product meets RoHS rules on harmful substances.

Manufacturing_Date: Date on the label is read with OCR and stored in the DD-MM-YYYY format we prefer.

OCR_Text: Full text pulled from the label by the OCR , covering date, compliance, serial number, and more.

QR_Data: Information read from the QR code, usually listing the Batch ID , Serial Number, Manufacturing Date etc.

Product_Image: A color photo of the product taken by webcam or fetched from a dataset for defect checks.

Label_Image: Close-up shot of the printed label used later for both OCR work and QR code reading.

Extracted_Label_Data: Results from OCR and QR scanning are blended together to see if they match what we expect.

True_Label: Ground-truth classification of the item, marked as either good or defective for final checks.

Predicted_Label: Outcome from our CNN defect model, labeling the item again as either good or defective.

OCR_Result_Log: Permanent record of every value the OCR found, kept for review and debugging.

11.Conclusion

This whole project delivers a smart and automated product labeling system that combines a stack of OCR, QR code decoding, and CNN-based defect detection technologies to ensure the best accurate traceability and quality control in manufacturing of a product. By extracting key data and detecting anomalies in real time of the products, the system improves efficiency, reduces manual errors, and supports compliance in total performance. It offers a scalable solution aligned with Industry goals, with potential for future integration into fully intelligent production lines.

12.References

- **ResNet (Deep Residual Learning):** He, K., Zhang, X., Ren, S., & Sun, J. (2016). *Deep Residual Learning for Image Recognition*.

  https://arxiv.org/abs/1512.03385

- **MVTec AD – Anomaly Detection Dataset**
  Bergmann, P., Fauser, M., Sattlegger, D., & Steger, C. (2019).
  https://www.kaggle.com/datasets/thtuan/mvtecad-mvtec-anomaly-detection/data
- **QR Code Standard (Overview & Tools):**

https://www.qrcode.com/en/about/

- ISO/IEC 18004:2015 (QR Code Standard) – *Full text behind paywall*, but summaries available.Towards Data Science (Medium): Radu Floricica

  https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939

- **Object Defect Detection and Classification Using CNN**

  Vignan's Foundation for Science, Technology & Research -Using CNN (Sivaji et al.)
  https://vignan.ac.in/aqardownload/abet/Object%20Defect%20Detection%20and%20Classification%20Using%20CNN_sivaji.pdf